Silja Karesto

WebP-kuvaformaatin käyttö ja hyödyllisyys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

6.5.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Silja Karesto WebP-kuvaformaatin käyttö ja hyödyllisyys 39 sivua 6.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Lehtori Jonna Eriksson

Insinöörityön tarkoituksena oli tutkia WebP-kuvaformaattia, jonka kehittäjä lupaa sen pakkaavan kuvia tehokkaammin kuin nykyiset yleisessä verkkokäytössä olevat kuvaformaatit. WebP tukee häviöllistä ja häviötöntä pakkaamista sekä animaatiota.

Työn tavoitteena oli selvittää, miten WebP vertautuu insinöörityössä valittuihin kuvaformaatteihin, jotka olivat JPG, PNG ja GIF. Lisäksi selvitettiin, vastaako pakkauksen teho väitteitä. Tehoa tutkittiin tarkkailemalla syntyvän tiedoston kokoa ja objektiivisesti mitattavaa laatua. Työn tavoitteena oli myös, että työn tulokset olisivat hyödynnettävissä osana verkkosivujen sisällön optimointia. Insinöörityö tehtiin omasta mielenkiinnosta aiheeseen, eikä sillä ollut tilaajaa.

Työssä perehdyttiin ensin siihen, mitä ominaisuuksia digitaalisella kuvalla on, sekä digitaalisten kuvien eri pakkausmenetelmiin ja valittuihin kuvaformaatteihin. Tämän jälkeen työssä perehdyttiin WebP-formaatin pakkausteknologiaan ja siihen, miten sitä käytetään. Lisäksi tarkasteltiin WebP-formaatin selain- ja käyttöliittymätukea ja kehityshistoriaa. Näiden pohjalta arvioitiin mahdollisia tulevaisuuden näkymiä. Kuvaformaatilla on prosentuaalisesti hyvä tuki kaikista selaimista tarkasteltuna, mutta sen tuki puuttuu useasta suuresta selaimesta, joten on vaikeaa arvioida sen käyttöä tulevaisuudessa.

Työ toteutettiin kahdessa osassa. Ensimmäisessä osassa tutkittiin WebP-kuvaformaatin kehittäjän tuloksia sen pakkauskyvystä ja laadusta tekemällä sen eri ominaisuuksista vertailuja valittuihin kuvaformaatteihin. Häviöllistä pakkausta verrattiin JPG-formaatin pakkaukseen, häviötöntä pakkausta PNG-formaatin pakkaukseen ja animaation pakkausta GIF-formaatin pakkaukseen. Kuvaformaattien vertailu osoitti, että WebP-formaatti pakkaa kuvat tasapuolisessa vertailussa valittuja kuvaformaatteja pienemmiksi ja säilyttää vastaavan laadun. Laatu todettiin käyttäen matemaattista menetelmää, jolla saatiin visuaalinen ja numeerinen tulos. Lisäksi WebP-formaatti pienentää JPG-pakkausta entisestään, mutta säilyttää kuitenkin hyväksyttävän kuvanlaadun. WebP-kuvaformaatti on varteenotettava tapa pakata kuvia verkkoon. Insinöörityössä toteutettua kuvagalleriaa esimerkkinä käyttäen osoitettiin pienemmäksi pakattujen kuvien vaikutus verkkosivujen latausajan nopeutumiseen.

Avainsanat	WebP, kuvaformaatit, kuvanpakkaus, sisällön optimointi

Author	Silja Karesto
Title	Usage and benefit of the WebP image format
Number of Pages	39 pages
Date	6 May 2017
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Jonna Eriksson, Senior Lecturer

The objective of the thesis was to study WebP, an image format which is said to compress images more effectively and with better quality than the most used image formats on the web. WebP image format supports both lossy and lossless image compression as well as animation. The aim was to examine how WebP compares with the image formats chosen for this thesis, which were JPG, PNG and GIF. Furthermore, this study evaluates if the WebP image format's compression is as good as its developers claim.

Though the thesis was conducted of personal interest, and does not have a client, the results of the thesis should be beneficial to the optimization of websites. The subject was approached by examining what the properties of digital photographs are and in what different ways it can be compressed. Also, the selected image formats were examined. Next the use and compression technique of the WebP image format were examined. Its current browser and operating system support were examined and analysis on its possible future was carried out.

WebP is a considerable option to compress images for the web, but even though it has good browser support measured in percentages, it still lacks support from major browsers making it difficult to predict its usability in the future.

The work was done in two parts, first comparing the compression on WebP to the selected other image formats, so that lossy WebP was compared to JPG, lossless WebP compared to PNG and animated WebP compared to GIF. The comparison of the image formats showed that in a fair comparison WebP compresses the images more while preserving better quality, than the selected image formats. The quality was visually and numerically determined using a mathematical method to measure the perceived quality. Additionally, WebP further compresses a JPG image still preserving acceptable quality. The impact of WebP compression on website load time was determined using an online image gallery, which was created for the thesis as an example. According to the results, WebP lessened the page load time due to its more compressed size.

Keywords	WebP, image formats, image compression, optimization

Sisällys

Lyhenteet

JPG	*Joint Photographic Experts Group*. Häviöllisesti pakkaava kuvaformaatti.
PNG	*Portable Network Graphics*. Häviöttömästi pakkaava kuvaformaatti.
GIF	Graphic Interchange Format. Häviöttömästi pakkaava kuvaformaatti. Tukee animaatiota.
RLE	*Run-length*. Algoritmi ryhmittää kuvassa esiintyviä kirkkausjaksoja koodilla, joka kertoo kirkkausarvon ja jakson pituuden.
LZW	*Lempel-Ziv-Welch*. Häviötön pakkausalgoritmi.
LZ77	*Lempel-Ziv*. Häviötön pakkausalgoritmi.
DCT	*Discrete Cosine Transformation*. Käytetään muuntamaan kuva kuva-avaruudesta taajuusavaruuteen.
RGB	*Red Green Blue*. Väritila, joka koostuu punaisesta, sinisestä ja vihreästä valosta.
RIFF	*Resource Interchange File Format*. Toimii säiliönä ei tiedostotyypeille.
SSIM	*Structural Similarity*. Matemaattinen keino esittää kuvan rakenteellista muutosta.
PPI	*Pixels Per Inch*. Ilmaisee, kuinka monta kuvapikseliä on tuumaa kohden.
DPI	*Dots Per Inch*. Ilmaisee, kuinka monta tulostuspistettä on tuumaa kohden.
NEF	*Nikon Electronic Format*. Nikon-kameravalmistajan oma RAW-formaatti.
HDR	*High Dynamic Range*. Kuvassa on korkea dynamiikka.
KiB	*Kibitavu*. Binäärijärjestelmän mukaisesti 2^{10} eli 1024 tavua.

1 Johdanto

Insinöörityössä tutkitaan Googlen kehittämää WebP-kuvaformaattia, joka tukee häviötöntä ja häviöllistä pakkausta sekä animaatiota. Googlen mukaan WebP pakkaa kuvat huomattavasti pienemmäksi kuin yleisimmät vastaavat kuvaformaatit vaikuttamatta kuitenkaan havaittavasti kuvan laatuun.

Työssä perehdytään digitaalisiin kuviin käyden läpi niiden rakennetta, eri vaikuttavia muuttujia, kuten laatu ja koko, sekä valittuja yleisimpiä kuvaformaatteja. Valitut kuvaformaatit ovat häviöllisesti pakkaava formaatti JPG, häviöttömästi pakkaava formaatti PNG ja animaatiota tukeva formaatti GIF. Lisäksi työssä käsitellään verkkosivun optimointia sisällön näkökulmasta.

WebP-formaatti sisältää kaikkia kolmea valittua vertailukuvaformaattia vastaavia ominaisuuksia. Työssä pyritään vertaamaan näitä ominaisuuksia vastaaviin kuvaformaatteihin ja mittaamaan mahdollinen hyöty. Mittaamisessa käytetään SSIM-indeksiä, eli matemaattista keinoa mitata rakenteellista muutosta kuvassa, vertaamaan vaikutusta laatuun häviöllisen pakkauksen suhteen, ja lisäksi verrataan pakattujen kuvatiedostojen kokoa.

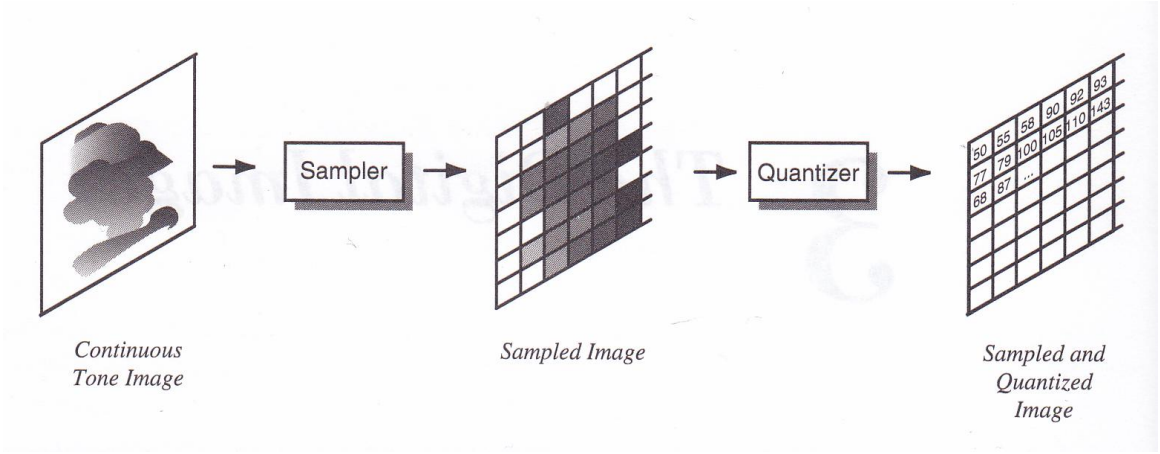
Kuvaformaattien verkkosivuilla käytön vertailuun käytetään esimerkkiympäristönä yksinkertaista kuvagalleriaa. Työssä käytetään kahta eri optimointityökalua mittaamaan kuvan pakkaamisen tuomaa hyötyä. Pakkaamisen vaikutusta tarkastellaan myös Chrome-verkkoselaimen kehittäjän työkalun avulla vertaamalla pakkaamisen vaikutusta kuvien latausaikaan.

Työn tavoitteena on selvittää, miten WebP-formaatti eroaa olemassa olevista yleisessä käytössä olevista formaateista ja onko sen hyöty yhtä suuri, kuin Google on omilla tutkimuksillaan selvittänyt.

2 Digitaaliset kuvat

Digitaalinen kuva voi olla bittikarttakuva, joka koostuu joukosta pikseleitä, tai vektorikuva, joka koostuu joukosta matemaattisesti määriteltyjä objekteja (1, s.18–19). Tässä työssä digitaalisella kuvalla viitataan bittikarttakuvaan.

Digitaalinen kuva on numeerinen esitys kuvasta, jota tarkastellaan tietokoneen tai mobiililaitteen näytöllä. Se muodostuu matriisista pikseleitä, joista jokainen on itsenäinen, ympäröivistä pikseleistä riippumaton. (1, s.18.) Digitaalisen kuvan muodostamiseksi täytyy analoginen signaali muuntaa digitaaliseksi. Sitä varten analogisesta signaalista otetaan näytteitä matriisin muodostamiseksi, ja kuvan tapauksessa näytteenottotaajuus määrittää kuvan tarkkuuden. (2, s. 38–39.) Kvantisoimalla näytteet määritetään jokaiselle näytteelle numeerinen arvo, joka kertoo pikselin kirkkausarvon. Pikselin väriarvo koostuu kolmen värikomponentin, eli punaisen, sinisen ja vihreän, kirkkausarvosta. (3, s. 38.) Tätä prosessia kutsutaan kuvan digitoinniksi, ja se on esitetty kuvassa 1.



Kuva 1. Kuvan digitointiprosessi (3, s. 38).

2.1 Kuvan koko

Kuvan koko voidaan käsittää usealla tavalla. Tässä työssä kuvatiedoston koko tarkoittaa sitä, kuinka monta bittiä tiedostoon pakattu kuva on. Kuvan koko pikseleinä puolestaan tarkoittaa, kuinka monesta pikselistä kuva koostuu. Kuvan fyysinen koko ottaa huomioon resoluution ja kertoo, minkä kokoinen kuva fyysisesti on.

Mitä suurempi kuvan koko on pikseleinä, sitä suurempi on myös kuvatiedoston koko, sillä sitä enemmän yksityiskohtia ja värejä kuva sisältää. Pikselien määrä vaikuttaa kuvatiedoston kokoon siten, että jokainen kuvan pikseli sisältää kuvan muista pikseleistä riippumatonta väritietoa. (1, s. 18.) Väritiedon määrä määrittää pikselin koon bitteinä, ja mitä suurempi pikselin bittisyys eli värisyvyys on, sitä enemmän siihen mahtuu värivaihtoehtoja. Pikseli, jonka koko on yksi bitti, voi olla vain kaksi arvoa, joten se voi sisältää enintään kaksi sävyä. Pikselillä, joka on kooltaan kahdeksan bittiä, on 2^8 eli 256 mahdollista värivaihtoehtoa. Tällaiset kuvat ovat harmaasävykuvia. 24-bittisissä RGB-väriavaruuden kuvissa on kahdeksan bittiä jokaista kolmea värikanavaa kohden, ja niistä kukin voi sisältää 256 eri sävyä. Tämä mahdollistaa 16 777 216 eri väriä laskentakaavalla 2^{24} tai 256^3. Jos RGB-kuvan värisyvyys on 32 bittiä, se tarkoittaa, että pikselin värikomponenttien lisäksi käytössä on ylimääräinen kahdeksan bittiä kuvaamaan pikselin alpha-kanavaa eli läpinäkyvyyttä. Kuvaa, jonka pikselit sisältävät tämän 24-bittisen tai 32-bittisen värimäärän, kutsutaan yleensä täysvärikuviksi. (2, s. 36–37; 4.)

Pikseleillä ei ole fyysistä kokoa, joten resoluutiota käytetään kertomaan, kuinka monta pikseliä on tuumalla, jotta saadaan määritettyä kuvalle fyysinen koko eli tulostuskoko. Resoluutio, eli pikselitiheys, määrittää, minkä kokoinen kuva on luonnossa, ja sitä ilmaisemaan käytetään yksikköä ppi, joka tulee sanoista pixels per inch eli pikseliä tuumaa kohti. Ppi on eri asia kuin dpi, joka tulee sanoista dots per inch. Se on tulostamisessa käytetty yksikkö, ja kertoo, kuinka monta tulostepistettä tulee tuumaa kohden. (1, s. 21–22.) Pikselitiheys ei vaikuta kuvatiedoston kokoon, eikä se vaikuta kuvan laatuun näytöllä, mutta se vaikuttaa kuvan laatuun tulostettaessa. Selaimessa katsottuna kuvan koko pysyy samana pikselitiheydestä riippumatta, sillä selain, jolla kuvaa katsotaan, katsoo vain kuvan kokoa pikseleinä. (5.)

2.2 Kuvan laatu

Rasterikuvan kuvanlaadun määrittää kaksi tekijää: kuvan koko pikseleinä ja kuvan värisyvyys. Kuvan koko pikseleinä tarkoittaa, kuinka monta pikseliä kuvassa on vaakasuunnassa ja pystysuunnassa. Värisyvyys puolestaan tarkoittaa, kuinka monta bittiä käytetään yhden pikselin esittämiseen. Mitä enemmän kuvassa on pikseleitä, sitä tarkemmin siinä saadaan esitettyä yksityiskohtia, ja mitä suurempi värisyvyys kuvassa on, sitä useampia eri värejä kuvan esittämiseen voidaan käyttää.

Heikko kuvanlaatu, eli kuvan informaation vähäisyys, ilmenee kuvassa kuva-artefakteina. Kaareville tai diagonaaleille reunoille ilmestyy laadun heikentyessä porrasmaista artefaktia, mikä johtuu häviöllisten pakkausalgoritmien lohkoittain tapahtuvasta koodauksesta. Samasta syystä kuva-alueille, joissa on tarkkoja voimakkaan kontrastin reunoja, ilmestyy ringing-artefaktiksi kutsuttua kuvahäiriötä. Siinä reunan ympärille ilmestyy reunan muotoisia mustavalkoisia viivoja. Liukuväreihin ja muille alueille, joilla ilmenee loivaa sävyn muutosta, ilmaantuu sävyjen porrastusta. (6.)

Kuvassa 2 on havainnollistettu, miten ilmapallon reunaan ilmestyy häviöllisen pakkauksen yhteydessä häiriötä, joka on laatikkomaista ja esiintyy voimakkaan sävyeron yhteydessä. Myös kuvan taivasalueella näkyy portaittaista sävyn muutosta verrattuna vasemmalla puolella olevaan kuvan hyvälaatuiseen versioon. Kuvaa pakatessa käytettävän värin kvantisoinnin vuoksi värisyvyys heikkenee eli kuvasta vähenee sävyjä eikä se enää riitä esittämään asteittain tapahtuvaa siirtymistä sävystä toiseen. Tällöin syntyy portaittaisia isompia yhden sävyn alueita, joissa on silmin havaittava sävyero naapurialueeseen. (3, s. 48–49.) Harmaissa sävyissä sävyeroa voi vielä entisestään korostaa niin kutsuttu Mach band -illuusio, jossa vierekkäin olevissa sävyissä tummempi sävy vaikuttaa todellista tummemmalta vaaleampaa sävyä vasten ja vaaleampi todellista vaaleammalta tummempaa sävyä vasten (3, s. 19).

Kuva 2. Vasemmalla olevan alkuperäisen kuvan häviöllisen tallentamisen jälkeen näkyy suurennoksessa oikealla pakkaamisesta johtuvaa laadun heikentymää.

2.3 Kuvan pakkaaminen

Kuvan pakkaamisessa eli kompressoinnissa on kyse kuvatiedoston koon pienentämisestä matemaattista algoritmia käyttäen. Kuvan pakkaus voi olla häviöllistä tai häviötöntä. (1, s. 27–28.) Digitaalisen kuvan pakkaaminen on välttämätöntä, sillä muuten tiedostojen koot kasvavat liian suuriksi, että niitä voitaisiin tehokkaasti välittää verkossa. Yhden täysvärikuvan pikselin koko on 24 bittiä eli 3 tavua. Esimerkissä, jossa on 8 megapikselin kuva, on 3264 x 2448 eli 7 990 272 pikseliä. Kuvan leveys- ja pituussuhde voi vaihdella kuvan lähteestä riippuen, mutta pikselien kokonaismäärä pyöristettynä lähimpään miljoonaan on kuvan koko megapikseleinä ilmoitettuna. Kun jokainen pikseli on 3 tavua, on kuvan koko pakkaamattomana noin 24 megatavua. (7, s. 525–526; 8.)

Kuvan pakkaaminen vähentää kuvan esittämiseen tarvittavan datan määrää, sillä yleensä kuvat sisältävät jonkin verran toistuvaa tai tarpeetonta, eli redundanttia, informaatiota (3, s. 28). Kuvan kompressoinnin tavoitteena on tunnistaa turhat toistuvuudet ja koodata ne uuteen, vähemmän dataa vaativaan muotoon. Koska olisi laskennallisesti hidasta ja raskasta tunnistaa jokaisen kuvan redundantti data ja löytää paras mahdollinen tapa pakata se, on kehitetty eri pakkaustekniikoita, jotka kohdentuvat pääasiassa tietyille kuvatyypeille tai pakkausvaatimuksille. Yhden kuvanpakkaustekniikan toistaminen usean kertaa samalle kuvalle ei välttämättä pienennä kuvatiedoston kokoa enää entisestään, sillä redundantti data on jo pakattu. Usean eri pakkaustekniikan kohdistaminen samaan kuvaan voi kuitenkin johtaa pakatumpaan muotoon, jos ne kohdistavat pakkauksen erityyppiseen redundanttiin dataan. (3, s. 182.)

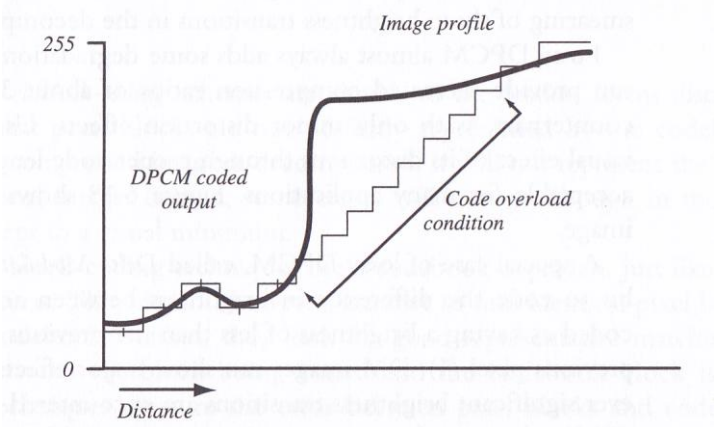
Kaikki kuvan pakkausjärjestelmät ovat kaksiosaisia: ne sisältävät kuvan tiedon pakkausoperaation ja käänteisen tiedon purkuoperaation. Pakkausoperaatio muuntaa alkuperäisen datan pakattuun datamuotoon, ja purkuoperaatio muuntaa pakatun datan takaisin alkuperäiseen pakkaamattomaan muotoon. (3, s. 180–181.)

2.3.1 Häviöllinen pakkaaminen

Häviöllisessä kuvan pakkaamisessa kuvasta sananmukaisesti häviää informaatiota. Siinä muodostetaan alkuperäisestä kuvasta uusi kuva, jossa kuvan yksityiskohtien ja värien määrää on vähennetty. Tällöin kuva voidaan muodostaa vähemmillä biteillä. (1, s. 27.)

Häviöllinen pakkaaminen on peruuttamatonta, eli pakkaamisen jälkeen kuvasta on hävinnyt dataa eikä sitä voida enää palauttaa (1, s. 27). Laadukkaassa pakkaustekniikassa kuvan kokoa saadaan pienennettyä paljonkin, ennen kuin se alkaa näkyvästi vaikuttaa kuvan laatuun (4). Häviöllisen pakkauksen etu on sen kyky pakata kuva paljon pienempään kokoon kuin häviöttömän pakkauksen. Kaikki häviölliset pakkaustekniikat poistavat dataa, mutta kaikissa tapauksissa dataa ei poisteta suoraan kuvasta. Sen sijaan kuva muutetaan erilaiseen muotoon, josta sitten poistetaan osia. Eri häviölliset pakkaustekniikat eroavat tavassa toteuttaa kuvan muuttaminen ja datan poistaminen. (3, s. 200.)

Häviöllistä pakkausta voi tehdä ennakoivalla koodauksella, jossa pikselien kirkkauksia käydään läpi ja edellisen pikselin arvon perusteella ennakoidaan seuraavan pikselin arvo. Tällöin koodataan vain kahden pikselin erotus. Yleinen muoto tästä on DCPM-pakkaus, joka tulee sanoista differential pulse code modulation. Pakkausmenetelmä käy läpi kuvan pikselit yksi kerrallaan. Ensimmäisen pikselin kirkkaus koodataan muuttumattomana ja siirrytään seuraavaan pikseliin. Seuraavan pikselin kirkkaudesta vähennetään edeltävän pikselin kirkkaus, ja näiden erotus koodataan. Näin edetään koko kuvan läpi. Operaatio perustuu oletukseen, että pikselien arvot ovat lähellä toisiaan, jolloin niiden erotus on pieni. Pienempi erotus voidaan koodata vähempibittisenä. Mikäli kuvassa tulee vastaan voimakas kirkkausero, voi kirkkauksien erotuksen arvo olla suurempi kuin maksimi arvo, joka erotukselle voidaan antaa. Tällöin tallennetaan maksimiarvo, kunnes voidaan palata takaisin oikeisiin erotuksien arvoihin, kuten kuvassa 3 on esitetty. Tämä voi ilmetä kuvassa sotkuisuutena tällaisilla alueilla. (3, s. 196–198, 205.)



Kuva 3. Jos kuvassa tulee vastaan voimakas kirkkausero, käytetään maksimiarvoa, kunnes saadaan taas palattua oikeisiin arvoihin (3, s. 205).

Häviöllinen pakkaus voidaan toteuttaa lohkopakkaustekniikalla, jossa kuvan pikselit jaetaan erikseen käsiteltäviin lohkoihin eli blokkeihin. Yksi blokki voi olla kooltaan yhdestä yli sataan pikseliä, kuten esimerkiksi 8 x 8 tai 16 x 16 pikselin blokki. Blokin sisältä etsitään redundanttia informaatiota. Lohkopakkaus perustuu tilastolliseen todennäköisyyteen, että kirkkausjaksot toistuvat läpi kuvan. Lohkopakkauksessa käytetään sanastoa tai koodistoa, johon tallennetaan toistuvia kirkkausjaksoja. Tallennettuihin jaksoihin verrataan kuvan blokkeja, ja yhteensopivan löytyessä määrätään sille viittaus sanastoon. Sanasto on tiedostokohtainen. (3, s. 196–199.) Häviöllisessä lohkopakkauksessa algoritmi ei pyri löytämään täydellistä yhteensopivuutta, vaan mahdollisimman hyvää vastaavuutta. Häviöllinen lohkopakkaus voidaan jakaa edelleen vektorikvantisointipohjaiseen tekniikkaan ja fraktaalipohjaiseen tekniikkaan. Kuvan pakkauksessa käytetty sanasto tulee liittää mukaan pakattuun kuvatiedostoon, jotta sitä voidaan käyttää myöhemmässä vaiheessa kuvan pakkausta purettaessa. (3, s. 207–208.)

Transform coding eli muunnoskoodaus on yksi häviöllisen lohkopakkauksen muoto. Muunnoskoodaus ei käytä sanastoa lohkojen tallentamiseen, vaan lohkot muunnetaan kuva-avaruudesta taajuusavaruuteen. Kuvan taajuusavaruuden taajuuskomponenttien arvot tallennetaan pakattuun kuvaan. Tässä vaiheessa arvot, jotka ovat nolla tai lähes nolla, yleensä poistetaan, sillä niillä on vähän arvoa kuvan esityksen kannalta. Kun pakkaus puretaan, kuva käänteisesti muunnetaan taajuusavaruudesta takaisin kuva-avaruuteen. (3, s. 209.)

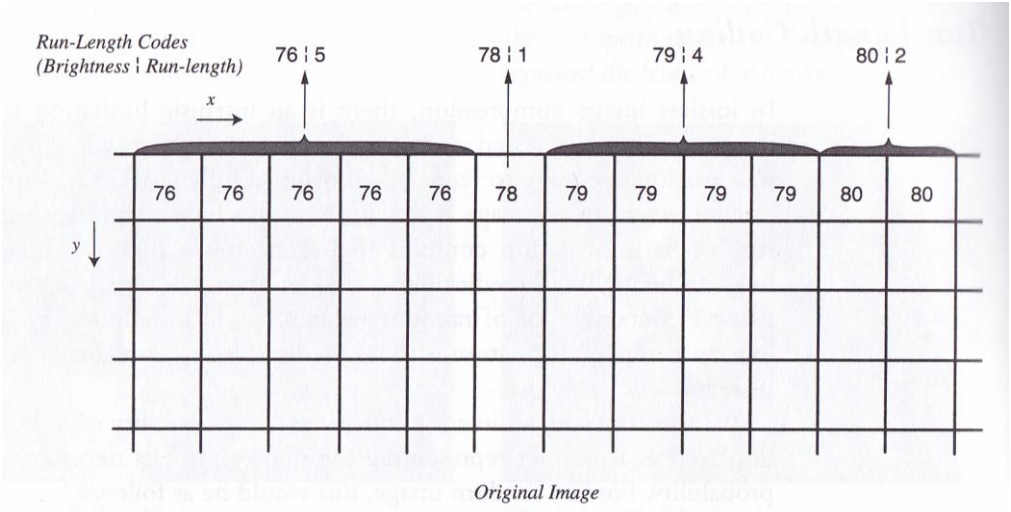
Informaation vähenemisen vuoksi häviöllinen pakkaaminen soveltuu parhaiten valokuville, joissa on lähtökohtaisesti paljon yksityiskohtia ja värejä, jolloin ihmissilmä ei niin helposti erota laadun heikentymistä huonon erotuskykynsä vuoksi (1, s. 27–28).

2.3.2 Häviötön pakkaaminen

Häviöttömässä kuvan pakkaamisessa kuvan alkuperäinen data on mahdollista palauttaa pakkauksen jälkeen, eikä se heikennä kuvan laatua. Häviöttömässä pakkauksessa kartoitetaan kuvasta alueita, joissa väriarvot ovat yhteneväisiä ja jotka voidaan siksi ilmoittaa vähemmillä biteillä. (1, s. 28.) Häviöttömällä pakkauksella ei saada aikaiseksi yhtä hyvää pakkaussuhdetta kuin häviöllisellä, mutta sitä on hyvä käyttää tilanteissa, joissa kuvan tieto tulee säilyttää alkuperäisenä tai on vaara, että hienovaraisista yksityiskohdista häviää tärkeää tietoa (3, s. 29).

Kuvan entropialla mitataan kuvan sisältämän informaation määrää. Jos kuvan entropia on korkea, sen sisältämä informaatio on vaikeasti ennustettavaa, eli se sisältää paljon irrallista informaatiota eikä juurikaan turhaa tai ylimääräistä. Alhaisen entropian kuvassa on vähän irrallista informaatiota ja paljon redundanssia. Entropiakoodaus hyödyntää tätä tietoa, ja entropiakoodaukseen perustuu esimerkiksi Huffman-koodaus. Siinä pikselin väriarvot nimitetään uudelleen vaihtuvamittaisiin koodeihin perustuen niiden toistuvuuden määrään. Usein toistuvat arvot esitetään lyhyemmällä koodilla ja harvemmin toistuvat pidemmällä, jolloin pakatun kuvan esittämiseen tarvitaan vähemmän bittejä. (3, s. 189–193.)

Toinen yleinen häviötön kuvanpakkaustekniikka on RLE eli run-length-koodaus, joka hyödyntää tietoa, että paikoittain lähekkäisillä pikseleillä on todennäköisesti sama kirkkausarvo. Se etsii kuvasta näitä jaksoja pikseleitä ja ryhmittää ne koodilla, joka koostuu pikselin kirkkausarvosta ja jakson pituudesta. (3, s. 189–191.) Tämä on havainnollistettu kuvassa 4.



Kuva 4. Kuvan viidellä ensimmäisellä pikselillä on kirkkausarvo 76, joten ne korvataan koodilla 76 | 5 (3, s. 190).

Häviötöntä pakkausta voi tehdä myös ennakoivalla koodauksella. Se eroaa häviöllisestä ennakoivasta koodauksesta suurien erotusarvojen koodauksen osalta. Jos vierekkäisten pikselien kirkkauden erotus on suurempi kuin suurin arvo, mikä niille voidaan tallentaa, ne koodataan suurempibittisenä, jolloin arvot tallennetaan oikein. (3, s. 196–198.)

Häviötöntä pakkausta voidaan lisäksi tehdä lohkopakkauksena, kuten häviöllistäkin pakkausta. Häviötön lohkopakkaus poikkeaa häviöllisestä siten, että kirkkausjaksolle sanastosta etsittävän viittauksen tulee olla täydellinen vastaavuus. (3, s. 196–199.)

2.4 Verkossa yleiset kuvaformaatit

Verkossa ovat vakiintuneet käyttöön tietyt kuvaformaatit: JPG, PNG ja GIF (9). Verkossa käytetyillä formaateilla on kaksiosainen tunniste, jota kutsutaan mediatyypiksi. Mediatyypistä käytetään myös nimitystä MIME-tyyppi (Multipurpose Internet Mail Extension). Mediatyyppi kertoo välitetyn sisällön tai datan yleisen tyypin ja tiedostomuodon. Mediatyyppejä standardisoi ja julkaisee IANA eli Internet Assigned Numbers Authority. Mediatyyppi koostuu kahdesta osasta ja valinnaisista parametreista. Ensimmäinen osa on varsinainen tyyppi, ja toinen osa on alatyyppi. Päätason tyyppejä ei ole monta. Niitä ovat muun muassa fontti, kuva, teksti tai video. Alatyyppi kuvaa mediatyyppiä tarkemmin ja sisältää usein sen nimen. (10.)

Tässä luvussa käsiteltävien yleisten kuvaformaattien mediatyypit ovat seuraavanlaiset:

- JPG-formaatin mediatyyppi on image/JPG.
- PNG-formaatin mediatyyppi on image/png.
- GIF-formaatin mediatyyppi on image/gif.

Päätason tyyppi ja alatason tyyppi erotetaan toisistaan vinoviivalla. Googlen kehittämälle WebP-kuvaformaatille on valittu mediatyyppi image/webp, mutta sitä ei toistaiseksi ole rekisteröity IANA:n viralliselle listalle (11). Mozillan, Firefox-selaimen kehittäneen yrityksen, ohjelmistokehittäjille tarkoitetuilla sivuilla mediatyyppi on lueteltu (10).

2.4.1 JPG-kuvaformaatti

JPG-lyhenne tulee sanoista Joint Photographic Experts Group, ja siitä käytetään yleensä tiedostopäätettä .jpg tai .jpeg. Päätteiden välillä ei ole juurikaan eroa, mutta JPG on yleisemmin käytössä. (12.)

JPG on häviöllisesti pakkaava formaatti, mutta pakkauksen määrää voi säätää, jolloin voi tasapainotella kuvan laadun ja tiedoston koon välillä. JPG-kuvat tukevat 16,7:ää

miljoonaa väriä, eli ne ovat täysvärikuvia, ja siistä syystä ne soveltuvat hyvin paljon yksityiskohtia sisältäville kuville, kuten valokuville, sekä kuville, joissa on tasaisia variaatioita sävyjen ja värien välillä. JPG-pakkaus hyödyntää ihmissilmän puutteita värien tummuusasteiden erottelukyvyssä ja poistaa pakatessa vähiten erottuvaa tietoa. Ihmissilmä havaitsee herkemmin kirkkauden kuin värin vaihteluita, joten värin vaihtelua yleensä poistetaan enemmän kuin kirkkauden vaihtelua. JPG ei sovellu hyvin kuviin, joissa on jyrkkiä kontrasteja reunojen välillä, sillä silloin häviöllisen pakkauksen vuoksi syntyy helposti virheitä kuvaan. (2, s. 60–61.)

JPG käyttää lohkopakkausta kuvan pienentämisessä ja pienentää kuvaa vähentämällä siitä pieniä sävyjen vaihtelun yksityiskohtia. Se tapahtuu muuntamalla kuvatiedoston väriavaruutta, jossa jokaista pikselin kirkkautta ja väriä käsitellään erikseen lohkopakkauksen mukaisesti 8 x 8 pikselin lohkoina. Pikselien arvot muunnetaan taajuusfunktioksi Fourier-muunnoksella. Tällöin pienet kirkkauden ja värin eroja vastaavat korkeat taajuudet voidaan suodattaa pois. Mitä suurempi pakkausaste, sitä karkeammin suodattamalla tämä tehdään. (2, s. 61.)

Joint Photographic Experts Group on myös kehittänyt JPG2000-kuvaformaatin, jonka tiedostopääte on JP2. JPG-formaatista poiketen JPG2000 pohjautuu aallokemuunnosmenetelmään, joka ehkäisee JPG-pakkaukselle tyypillisiä laatikkomaisia artefakteja. JPG2000 tukee sekä häviöllistä että häviötöntä pakkausta. JPG2000-formaatti pakkaa kuvat paremmalla laadulla kuin JPG, mutta se ei siitä huolimatta ole yleistynyt käytössä. (13; 14.)

Toinen JPG-formaatti, jossa on molemmat häviöllinen ja häviötön pakkaus, on JPG-XR. Se poikkeaa alkuperäisestä JPG-formaatista muun muassa sen laajemman väritilojen tuen osalta. JPG-XR tukee RGB-väritilan lisäksi painotuotteissa käytettävää CMYK-väritilaa. Se tukee myös alpha-kanavaa, jota alkuperäinen JPG ei tue. Se on suunniteltu tukemaan HDR eli High Dynamic Range -kuvia, ja siinä on tehokkaampi pakkausalgoritmi. (15.) JPG-XR on tuettu vain Internet Explorer- ja Edge-selaimilla (16).

2.4.2 PNG-kuvaformaatti

Häviöttömästi pakkaava PNG kehitettiin vuonna 1995, ja vuonna 1996 siitä tuli World Wide Web Consortiumin suositus käytettäväksi kuville verkkosivuilla. PNG tulee sa-

noista Portable Network Graphics. PNG-formaatti luotiin GIF-formaatin seuraajaksi, kun syntyi tarve patentittomalle kuvaformaatille. Tarve syntyi, kun Unisys-yritys vahvisti patenttiaan GIF-formaatin käyttämään häviöttömään LZW-pakkausalgoritmiin. PNG-formaatin suosiota rajoitti aluksi Microsoftin Internet Exporer, IE, jonka aikaisemmat versiot eivät tukeneet kaikkia PNG-formaatin ominaisuuksia. Tämä korjaantui IE-versiossa 7. (17, s. 553, 559.) PNG ohitti GIF-formaatin suosituimpana häviöttömänä kuvaformaattina vuonna 2013 (18).

PNG-kuvaformaatti tukee 24-bittistä ja 48-bittistä täysväritallentamista, 16-bittistä harmaasävytallentamista ja väripalettipohjaista 8-bittistä indeksoitujen värien tallentamista. Vaikka PNG tukee yli 24-bittistä tallentamista, ei sitä ole tarkoitettu korvaamaan JPG-formaattia, sillä häviöttömänä sen tiedostokoko jää JPG-formaatin häviöllistä pakkausta merkittävästi suuremmaksi täysvärikuvilla. (17, s. 552–553.) Kaikilla mainituilla tallentamismuodolla voi olla alpha-kanava käytössä. PNG sallii jopa 254 eri läpinäkyvyysastetta, toisin kuin GIF, jossa on vain täysin läpinäkyvä ja täysin peitetty. Asteittainen läpinäkyvyys mahdollistaa myös antialiasoinnin, mikä mahdollistaa pehmeämmät reunat kuvalle. (19.)

Kuvan pakkaamiseen PNG-formaatti käyttää DEFLATE-pakkausmenetelmää. Se käyttää LZ77-algoritmin ja Huffman-koodauksen yhdistelmää. LZ77 on Lempel-Ziv sanastokoodaukseen perustuvan algoritmin versio, kuten myös GIF-formaatin käyttämä patentoitu LZW-algoritmi. Huffman-koodaus on puolestaan entropiakoodaustekniikka. (20; 3, s. 193, 199.)

PNG ei tue animaatiota, mutta vuonna 2001 PNG-formaatin kehittäjät julkaisivat MNG-formaatin, joka tukee animaatiota. MNG on lyhenne sanoista Multiple-image Network Graphics. (21.)

2.4.3 GIF-kuvaformaatti

GIF-lyhenne tulee sanoista Graphics Interchange Format, ja sen kehitti CompuServe-yritys vuonna 1987. GIF oli ensimmäinen värikuvaformaatti, joita ensimmäiset selaimet tukivat. (17, s. 529.)

GIF-kuvaformaatti tukee läpinäkyvyyttä ja soveltuu hyvin yksinkertaisille alhaisen resoluution kuville, joissa on vähän eri värejä, kuten logoille, symboleille ja muille kuvakkeil-

le. GIF-kuvaformaatti on rajoitettu enintään 256 eri väriin, ja tästä syystä se ei sovellu hyvin valokuville. (1, s. 28.) GIF-formaatti tukee myös yksinkertaista animaatiota ja kuvien esilataamista. Esiladattaessa kuva tulee ensin näkyviin epätarkkana ja kuvan latauduttua kokonaan se tarkentuu. Esilatausominaisuus kasvattaa hieman kuvan kokoa. (2, s. 62.) GIF tukee myös läpinäkyvyyttä, mutta toisin kuin PNG, se ei tue eriasteista läpinäkyvyyttä. Tämä voi aiheuttaa kuvaa pakattaessa kuvan ja läpinäkyvän taustan reunaan epätoivottua sädekehämäistä efektiä. Efekti syntyy, jos reunassa on käytetty antialiasointia eli sitä on pehmennetty. (17, s. 535.)

GIF-käyttää LZW-pakkausta, joka on häviötön lohkopakkaustekniikka (17, s. 529). Siinä sanasto alustetaan 256 yksittäisellä kirkkausarvolla, ja jokainen kuvan pikseli käydään läpi yksitellen siirtyen riviltä toiselle. Jos pikselin kirkkaus löytyy sanastosta, algoritmi lisää siihen seuraavan pikselin kirkkauden, muodostaen pikseleistä blokin. Jos kirkkauksien jaksoa ei löydy sanastosta, algoritmi lisää sen sanastoon. Jos jakso löytyy jo sanastosta, lisätään siihen seuraavan pikselin kirkkaus niin kauan, kunnes jaksoa ei löydy ja se lisätään uutena sanastoon. Lopulta algoritmi vertaa sanastoon tallennettuja pikselilohkoja kuvan pikselilohkoihin etsien pisintä mahdollista osumaa. Algoritmi korvaa löydetyn osuman pakattuun kuvaan viittauksella sanastoon. (3, s. 199–200.)

2.5 Verkkosivujen sisältöelementtien optimointi

Verkkosivujen sisältöelementtien optimointi on entistä tärkeämpää mobiililaitteiden käytön yleistyttyä. Mobiililaitteiden teho ei aina ole yhtä hyviä kuin tietokoneen, eivätkä yhteydet ole aina yhtä luotettavasti hyvät, joten optimointi on etenkin niiden kannalta tärkeää. (22.) Verkkosivuilla on monia tekijöitä, joissa on mahdollista vähentää ladattavan sisällön määrää, kuten pakkaamalla sivun esittämiseen käytettäviä HTML-, CSS- ja JavaScript-tiedostoja tai tallentamalla selaimen välimuistiin palvelimelta jo kertaalleen ladattua dataa. Samalla tavalla myös kuvien lataaminen voi vaikuttaa merkittävästi verkkosivujen latausaikaan, ja tästä syystä myös niiden pakkaaminen on tärkeää. (4; 23; 24.) Kuvat muodostavat jopa 60–65 % ladattavista biteistä suurimmalla osalla sivuista (25).

Verkkosivun latautuessa selain lähettää kyselyn palvelimelle, joka palauttaa pyydetyn datan (26). Jos pyydetyt tiedot, kuten esimerkiksi kuvatiedostot, ovat suuria, niiden latautuminen palvelimelta kestää ja pahimmillaan tämä hidastaa koko sivun latautumis-

ta. Verkkomarkkinointistrategioihin ja analytiikkaan keskittyneen Kissmetrics-sivuston tutkimuksen mukaan käyttäjät ovat valmiita odottamaan sivun latautumista tietokoneen selaimella noin kolme sekuntia, ja mobiililaitteen selaimella noin viisi sekuntia (27). Tästä syystä, jos kuvia saa pakattua mahdollisimman paljon pienemmiksi, se voi vaikuttaa latausajan lisäksi kävijämäärään merkittävästi, etenkin mitä enemmän sivulla on kuvia. Lisäksi Google on ilmoittanut käyttävänsä sivun latausnopeutta yhtenä tekijänä algoritmissa, jonka mukaan se järjestää hakutulokset (28).

3 WebP-kuvaformaatti

WebP-kuvaformaatti julkistettiin ensimmäisen kerran 30. syyskuuta 2010 uutena avoimena standardina täysvärikuvien näyttämiseen verkossa. WebP on Googlen kehittämä avoimen lähdekoodin kuvaformaatti, ja Googlen mukaan WebP pakatut kuvat ovat noin 26 % pienempiä kooltaan kuin PNG-kuvat ja 25–34 % pienempiä kuin JPG-kuvat vastaavalla kuvanlaadulla. WebP tukee sekä häviöllistä että häviötöntä kuvan pakkausta sekä läpinäkyvyyttä, animaatiota, metadataa ja väriprofiileja. WebP tarjoaa myös mahdollisuuden häviölliseen pakkaamiseen säilyttäen silti läpinäkyvyyden. WebP-kuvan maksimikoko pikseleinä on 16 383 x 16 383. (29; 30; 31; 32.)

3.1 WebP-pakkausmenetelmät

WebP-kuvaformaatin häviöllinen pakkaus hyödyntää samaa tekniikkaa kuin Googlen avoin videoformaatti WebM, jonka taustalla on VP8-videokoodekki. VP8-videonpakkaus on avoin formaatti, jonka On2 Technologies yhtiö on alun perin kehittänyt ja jonka Google osti vuonna 2010. (31; 32.) WebP-kuvaformaatin häviöllinen pakkaus perustuu VP8-videoformaatin yhden kuvakehyksen sisäisen datan ennustamiseen eli intra-frame-koodaukseen. Intra-frame-koodaus käyttää yhden kuvakehyksen dataa ja hyödyntää kuvan kuva-avaruuden redundanssia pikselien arvojen ennustamiseen. Tämä tarkoittaa, että on oletettavaa, että tietyn kuva-alueen eli lohkon sisällä olevien pikseleiden väriarvojen välillä on korrelaatiota, ja näin yksittäisen pikselin arvo on ennustettavissa, kun ympäröivien pikselien arvot tunnetaan. (31; 33.)

Häviöllinen pakkaus käyttää lohkoihin perustuvaa ennakoivaa koodausta pakkaukseen. Kuvakehys jaetaan pienempiin osiin eli lohkoihin, joita kutsutaan makroblokeiksi. Näis-

sä makroblokeissa algoritmi pyrkii ennakoimalla tunnistamaan redundanttia kirkkausinformaatiota perustuen aiempiin käsiteltyihin lohkoihin. Lohkojen lukemisjärjestys on vasemmalta oikealle ja ylhäältä alas. Tämän jälkeen redundantti informaatio voidaan vähentää lohkosta, jolloin jäljelle jää erotus. Vain erotus välitetään pakatussa muodossa. Jäljelle jääneelle erotukselle tehdään diskreetti kosinimuunnos, eli DCT, mikä tarkoittaa, että lohkon informaatiosta saadaan esiin kuva-avaruudelliset taajuuskomponentit. Taajuusmuunnoksen jälkeen jäävät DCT-kertoimet, jotka ovat pieniä arvoja tai nollia. Nämä arvot kvantisoidaan, mikä tarkoittaa, että arvot pyöristetään tiettyyn esitarkkuuteen arvoihin perustuvalla jakajalla. Tällöin korkeataajuisia komponentteja osoittavien kohtien arvot menevät nollaan ja matalataajuisia komponentteja osoittavien kohtien arvoiksi jää nollaa suurempi luku. Nämä arvot järjestetään siksak-kuvioisella skannauksella jonoon ja entropiakoodataan. Tällöin kuvasta on poistettu suuritaajuista informaatiota, jota ihmissilmän on vaikea erottaa. (31; 34; 35.)

Kvantisointi on häviöllisen WebP-koodauksen ainoa vaihe, jossa informaatiota peruuttamattomasti poistetaan (31).

Kuvanlaadun parantamiseksi kuvaa pakattaessa kuva jaetaan alueisiin, joilla on visuaalisesti samankaltaista sisältöä. Tällaisia alueita voivat olla esimerkiksi alueet, joilla on paljon informaatiota, ja alueet, joilla on vähän informaatiota. Näitä alueita voi olla yhdessä kuvassa enintään neljä, mikä johtuu VP8-koodekin rajoituksesta. Jokaiselle alueelle säädetään pakkaus parametrit erikseen, jotta saadaan tehokas pakkaus, jossa bitit on jaettu tasaisesti kuvan eri alueille siten, että korkean entropian alueilla on enemmän bittejä ja matalan entropian alueilla on vähemmän bittejä. (31.)

Häviöllisellä WebP-pakkauksella on samankaltaisuutta JPG-pakkauksen kanssa, mutta myös muutamia selviä eroavaisuuksia. WebP käyttää ennakointia lohkopakkauksessaan, toisin kuin JPG, ja Googlen mukaan tämä on suurin syy, miksi WebP onnistuu pakkaamaan kuvat tehokkaammin kuin JPG. Lisäksi WebP-pakkauksessa kvantisointi tehdään intra-frame-koodauksen ansiosta jokaiselle lohkolle erikseen, kun JPG-pakkauksessa samoja kvantisointikertoimia käytetään kaikille lohkoille. Entropiakoodauksessa WebP käyttää aritmeettista koodausta, mikä antaa 5–10 %:n pakkaushyödyn verrattuna JPG-formaatin käyttämään Huffman-koodaukseen. (31; 35.)

Häviöttömän pakkaustekniikan Googlen WebP-kehitysryhmä on kehittänyt itse. Se käyttää jo nähtyjä kuvan fragmentteja muodostaakseen uudet pikselit. Häviöttömässä

WebP-pakkauksessa kuva muunnetaan usealla tavalla. Ensin kuvaan käytetään ennakoivaa koodausta entropian vähentämiseksi ja vain erotus koodataan. Kuva jaetaan lohkoihin, ja samaa ennakointimoodia käytetään kaikkiin yhden lohkon pikseleihin. Seuraavaksi kuvan väriavaruus muunnetaan. Muunnoksessa pyritään poistamaan korrelaatio vihreän, punaisen ja sininen värikanavan välillä. Vihreä värikanava pysyy sellaisenaan ja punainen ja sininen muunnetaan sen perusteella. Häviötön WebP hyödyntää pakkauksessa jo käsiteltyjä lohkoja uusien muodostamisessa tallentamalla väliaikaiseen väripalettiin viimeisimpiä väriarvoja. Tämän jälkeen eri muunnoksien parametrit ja muunnettu kuvadata entropiakoodataan. Tähän koodekki käyttää LZ77:n ja Huffman-koodauksen yhdistelmän muunnosta, jota myös PNG käyttää. (31.)

3.2 Säiliömuoto

WebP-formaatti soveltaa RIFF-säiliömuotoa datan säilytykseen. RIFF on lyhenne sanoista resource interchange file format, ja se on Microsoftin kehittämä säiliömuoto. RIFF toimii yleisesti perustana eri tiedostotyypeille, eikä sillä ei ole omaa tiedostopäätettä. Se koostuu lohkoista, joihin tallennetaan eri tiedostotyyppien data. (36; 37.)

WebP-formaatin RIFF-säiliömuotoon perustuva säiliömuoto tarjoaa tuen WebP-formaatin eri ominaisuuksille, kuten häviölliselle ja häviöttömälle pakkaukselle, metadatalle, läpinäkyvyydelle, väriprofiileille ja animaatiolle. RIFF on kevyt säiliömuoto, joka tuo jokaiseen WebP-kuvaan vähintään 20 tavua lisää, mutta tallennettu metadata voi kasvattaa kokoa. (36; 37.)

3.3 WebP-kuvan konvertointi

Kuvien konvertoimiseksi WebP-formaattiin tarvitaan esikäännetty cwebp-konvertoijaohjelma. Googlen WebP-formaattia käsitteleviltä sivuilta voi ladata käyttöjärjestelmäkohtaisen paketin, joka sisältää libwebp-kirjaston sekä cwebp- ja dwebp-esikäännetyt ajotiedostot. Cwebp koodaa JPG-, PNG- tai TIFF-formaatin kuvat WebP-formaattiin, ja dwebp koodaa WebP-formaatista takaisin edellä mainittuihin formaatteihin. Paketti sisältää myös gif2webp-kääntäjän, jolla saa GIF-kuvan käännettyä WebP-kuvaksi. Paketit ovat olemassa Windows-, Linux- ja Max OS X -käyttöjärjestelmille. (38.)

Kun paketti on ladattu ja asennettu, voi kuvia konvertoida WebP-formaattiin käyttöjärjestelmän komentoriviltä. Esimerkiksi PNG-kuvan konvertointi WebP-formaattiin onnistuu komennolla, joka on esitetty esimerkkikoodissa 1.

```
cwebp esimerkkikuva.png -o esimerkkikuva.webp
```

Esimerkkikoodi 1. PNG-kuvan konvertointi WebP-kuvaksi.

Komentoon voi lisätä valintoja, esimerkiksi pakkauksen määrään tai tiedoston kokoon vaikuttavia. Pakkauksen määrään voi vaikuttaa antamalla pakkauskertoimen asteikolta 0–100, ja esimerkiksi JPG-formaatissa olevan valokuvan voi pakata 80 pakkauskertoimella esimerkkikoodissa 2 olevalla komennolla (39).

```
cwebp -preset photo -q 80 esimerkkikuva.jpg -o
esimerkkikuva.webp
```

Esimerkkikoodi 2. Valokuvatyyppisen JPG-kuvan konvertointi WebP-kuvaksi.

Komento on oletusarvoisesti häviöllisesti pakkaava. Häviöttömään pakkaukseen tulee lisätä valinta -lossless. Esimerkkikoodi 3 on esimerkki, miten konvertointi takaisin PNG-formaattiin onnistuu (39).

```
dwebp esimerkkikuva.webp -o esimerkkikuva.png
```

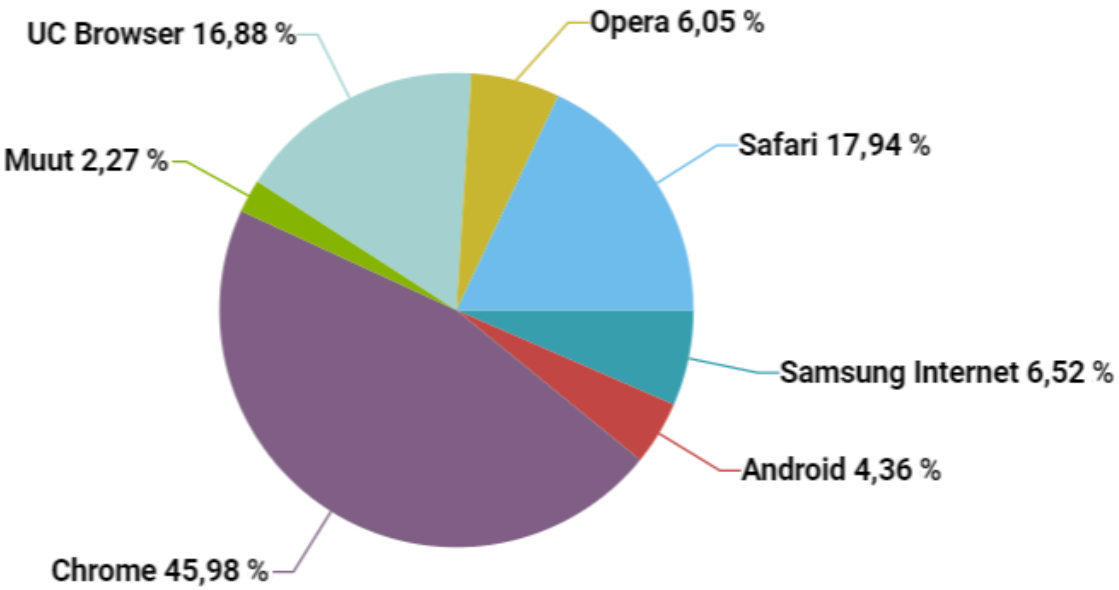
Esimerkkikoodi 3. WebP-kuvan konvertointi PNG-kuvaksi.

Joissain kuvankäsittelyohjelmissa on tuki WebP-kuvien tallentamiseen ja muokkaamiseen. Näitä ovat esimerkiksi Mac OS X -alustalla Pixelmator sekä ilmaiset kuvankäsittelyohjelmat ImageMagick ja GIMP. Lisäksi Adobe PhotoShop -kuvankäsittelyohjelmaan on mahdollista asentaa lisäosa, joka mahdollistaa WebP-formaattiin tallentamisen. Google tarjoaa ladattavan koodekin mukana ohjelmointirajapinnan ja dokumentaation, joiden avulla voi itse lisätä tuen WebP-formaatille sovellukseen. (39.)

3.4 Selaintuki ja käyttöliittymätuki

WebP-kuvaformaatin käytössä huomioitava ero muihin verkossa yleisesti käytettyihin kuvaformaatteihin verrattuna on sen puutteellinen tuki. WebP-kuvaformaatilla on toistaiseksi ominaisuuksista riippuen tuki Google Chrome -selaimella ja Opera-selaimella sekä Android-käyttöjärjestelmän natiiviselaimella (30). Safari- ja Firefox-selaimet kokeilevat tukea WebP-formaatille, mutta toistaiseksi sitä ei ole saatavilla. Lisäksi WebP on tuettu Aasian markkinoilla yleistyneessä UC Browser -mobiiliselaimessa. (40.)

Maailmanlaajuisesti verkkosivuja selataan enemmän mobiililaitteilla kuin tietokoneilla (41). Etenkin mobiililaitteille sisällön osalta optimoidut sivut ovat tärkeitä, sillä verkkoyhteydet voivat olla yhteyksien tyypistä riippuen hitaampia ja mobiililaitteilla on pienempi teho prosessoida ladattu sisältö (42). WebP-formaatti on tuettu Google Chrome -selaimella, Android Native -selaimella, UC Browser -selaimella, Samsung Internet -selaimella ja Opera-selaimella, jotka kattavat noin 70 % mobiilissa eniten käytetyistä selaimista. Kuvassa 5 on havainnollistettu näiden selaimien osuutta suhteessa kaikkiin selaimiin maailmanlaajuisesti maaliskuussa 2017. (40; 43.) WebP-formaatilla on siis hyvä tuki myös mobiiliselaimilla, mikä on tärkeää niiden lisääntyvän käytön vuoksi.



Kuva 5. Eri mobiiliselainten käyttö maailmanlaajuisesti maaliskuussa 2017 (43).

Mikäli selain ei tue WebP-formaattia, täytyy kuvalle olla varavaihtoehto selaimen tukemassa kuvaformaatissa, joka ladataan WebP-kuvan sijaan. Selaimen tuen voi tarkistaa joko palvelimen puolella tai asiakasohjelman puolella. Yksi keino on lähettää asiakasohjelmasta accept-tyyppinen request-header, jossa kerrotaan, minkätyyppiset tiedostot ovat hyväksyttyjä vastauksessa. Jos image/webp-mediatyyppi on hyväksytty, voi palvelin palauttaa WebP-kuvia. WebP-formaatin tuen voi myös tarkistaa selaimessa esimerkiksi JavaScript-kirjastolla. Yksinkertaistettuna JavaScriptillä tarkistetaan, onko WebP tuettu selaimessa, minkä perusteella tiedetään, mitä pyytää palvelimelta. (30.)

Käyttöjärjestelmillä ei toistaiseksi ole tukea WebP-formaatin näyttämiseen, mutta Windows-käyttöjärjestelmän käyttäjien on mahdollista asentaa WebP-koodekki Windowsiin. Tämä mahdollistaa sen, että WebP-tiedostoja voi avata Windows Photo Viewer -kuvankatseluohjelmalla ja että niiden esikatselukuvake on mahdollista nähdä Windows Explorerilla eli kansiossa. (44.)

3.5 Kehityshistoria ja tulevaisuuden kehitys

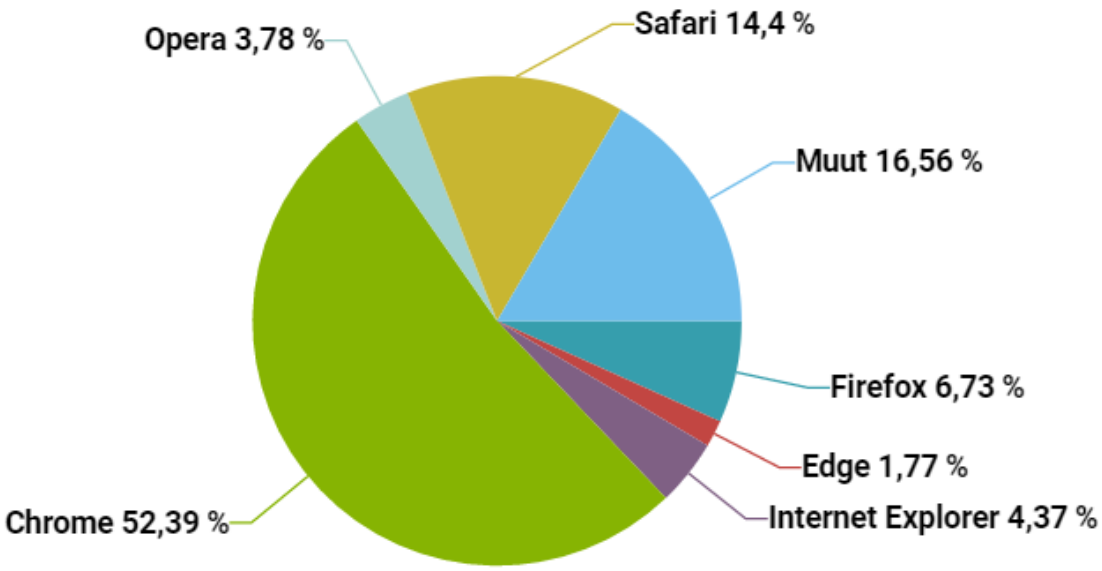
Googlen Chromium-blogissa on julkaistu tietoa WebP-formaatin uusista ominaisuuksista siitä lähtien, kun Google alkoi kehittää WebP-formaattiaan. Blogin WebP-formaattia käsittelevien artikkeleiden historiasta näkee, että Google on kehittänyt uusia ominaisuuksia WebP-formaattiin ja tehnyt niistä kertovia julkaisuja blogiin useita kertoja vuosien 2010 ja 2013 välillä. Uusin versiojulkaisu, eli libwebp 0.6.0, on julkaistu 31. tammikuuta 2017. Versiojulkaisuhistoriasta on nähtävissä, että uusia versioita on julkaistu vuosittain kahdesta kolmeen. (45.)

WebP-formaatin mahdollista tulevaisuuden kasvua ja laajentumista parantavat puheet saada formaatti tuetuksi Mozilla Firefox -selaimessa. Vuonna 2013 Mozilla antoi ymmärtää, että tukea WebP-formaattiin harkitaan (46). Vielä vuoden 2017 maaliskuussa tukea ei ole, mutta vuoden 2016 elokuussa WebP-formaatin tuen edistyminen Firefox-selaimelle nousi jälleen uutiseksi, ja senhetkisten arvioiden mukaan tuki on todella tulossa Firefox-selaimeen jossain tulevaisuuden päivityksessä. Tuki ei kuitenkaan olisi ainakaan alkuun automaattinen kaikille, vaan käyttäjien tulee itse käydä asettamassa se päälle selaimen asetuksissa. (47.)

Applen käyttöjärjestelmät ja Safari-selain eivät toistaiseksi tue natiivisti WebP-formaattia. Heinäkuussa 2016 uutisoitiin tuesta WebP-formaatille Safari-selaimen beetaversiossa iOS 10 ja MacOS Sierra -käyttöjärjestelmiin. (48.) Syyskuussa 2016 julkaistusta lopullisesta versiosta tuki on poistettu (40).

Selainten kehittäjien vastahakoisuuden lisätä WebP-formaatin tuki osaksi selaimiaan epäillään johtuvan päätöksen pitkän tähtäimen kannattavuudesta. Mikäli selainvalmistajat ottavat WebP-formaatin tuen osaksi selaimiaan, niiden pitää tukea sitä tulevaisuudessakin, vaikka formaatin suosio ei kasvaisi, se kasvaisi hitaasti tai formaatti jäisi kokonaan pois käytöstä. WebP-formaattia kohtaan on paljon kritiikkiä sen tarpeellisuudesta ja hyödyllisyydestä ja vaikutuksesta kuvan laatuun. (49.)

Googlen Chrome-selaimen osuus oli 52,39 % selainten käytöstä vuoden 2017 helmikuussa koko maailmassa ja kaikilla alustoilla. Firefox-selaimen osuus oli 6,73 %, Applen Safari-selaimen osuus 14,4 % ja Microsoftin Edge-selaimen 1,77 %. Kuvassa 6 havainnollistetussa tilastossa on otettu huomioon kaikki selaimet eli sekä työasemalla että mobiililaitteilla käytettävät selaimet. (50.) Tämän perusteella WebP-formaatilla on yksin Chrome-selaimen ansiosta vahva tuki, sillä Chrome on selvästi käytetyin selain, ja lisäksi ovat vielä Opera- ja Android-selainten osuudet. Maaliskuussa 2017 maailmanlaajuisesti 72,32 %:ssa selaimista on täysi tuki formaatille (39).



Kuva 6. Eri selainten käyttö kaikilla alustoilla maailmanlaajuisesti helmikuussa 2017 (50).

WebP-formaatin tuen saaminen muihinkin selaimiin on kuitenkin kehityksen kannalta tärkeää, sillä selainten valta-asemat muuttuvat koko ajan. Esimerkiksi vielä helmikuussa 2010 Interner Explorer oli selvästi suosituin selain maailmanlaajuisesti 53,56 %:n osuudella ja Firefox oli toiseksi suosituin 31,28 %:n osuudella ja Chromen osuus oli 6,6 %. Helmikuussa 2012 samat kolme selainta olivat enää 10 prosenttiyksikön sisällä toisistaan ja muodostivat 88,77 % kaikkien selainten käytön määrästä. Tästä eteenpäin Chrome-selain on tasaisesti kasvanut suosituimmaksi selaimeksi maailmanlaajuisesti, ja viimeisimmän vuoden tilaston mukaan sen osuus edelleen kasvaa tasaisesti, vaikka kilpailevien selainten määrä on kasvanut. (50.)

WebP tuo Googlen osoittamien testien perusteella olennaista hyötyä kuvien käyttöön verkossa. WebP-formaatilla on saatu pakattua huomattavasti pienempiä kuvia, mistä on etua verkkosivuja ladattaessa. Kuitenkin viimeisimmät suuret uutiset WebP-formaatin kehityksestä ovat vuodelta 2013, jolloin muun muassa tuki animoiduille WebP-kuville julkaistiin sekä muita parannuksia pakkaukseen. Formaatin yleistyminen tulevaisuudessa riippuu siitä, millaisen tuen eri selaimet uusissa versioissaan sille antavat ja miten niiden käsitteleminen eri ohjelmissa ja käyttöliittymissä monipuolistuu. Maaliskuussa 2017 WebP-kuvia käyttää alle 0,1 % kaikista verkkosivuista (51).

4 WebP-kuvaformaatin hyöty verrattuna perinteisiin kuvaformaatteihin

Insinöörityössä käsitellään JPG-, PNG- ja GIF-formaattia verrattuna WebP-formaattiin, jotta saadaan tutkittua WebP-formaatin mahdollista hyötyä ja tarpeellisuutta vallitsevien formaattien ohella. Työssä verrataan ensin jokaista formaattia erikseen, mutta asetetaan kuitenkin suurin painoarvo JPG-formaattiin. Vertailussa mitataan pakkauksen toimivuutta pitäen mielessä Googlen omat tulokset. Työn toisessa osassa verrataan JPG-formaattiin ja WebP-formaattiin pakattuja kuvia verkkosivuilla ja sitä, miten pakkaus mahdollisesti vaikuttaa sivun lataukseen.

4.1 Pakkauksen määrän määrittäminen

Kolmelle valitulle kuvaformaatille tehtiin erikseen vertailu, jossa niitä verrattiin vastaavaan konvertointiin WebP-formaattiin. Pakkauksesta pyrittiin tekemään mahdollisimman hyvä valitulle kuvalle ja toistamaan se vastaavilla asetuksilla WebP-formaatissa.

Näin voitiin verrata, kuinka paljon enemmän WebP pakkaa suhteessa verrattavaan kuvaformaattiin. Työssä kuvien käsittelyyn ja pakkaukseen käytettiin Adobe Photoshop -kuvankäsittelyohjelmaa ja Windows-käyttöjärjestelmän Command Prompt -komentotulkkia.

Häviöllisen pakkauksen suhteen verrattiin tiedoston koon lisäksi kuvanlaatua. Yksi keino alkuperäisen kuvan ja pakatun kuvan laadun vertailemiseen on käyttää kuvankäsittelyohjelmassa Difference-nimistä suodinta. Suotimen nimi ja käyttö saattaa vaihdella eri kuvankäsittelyohjelmien välillä. Kuvankäsittelyohjelmassa vertailtavat kuvat asetetaan päällekkäin erillisille tasoille niin, että alkuperäinen kuva on alla ja vertailtava kuva päällä. Päällä olevalle kuvalle laitetaan Difference-suodin, joka poistaa alla olevan kuvan pikselien eri värikanavien arvoista päällä olevan kuvan pikselien vastaavan värikanavan arvon. Jos värikanavien arvot ovat samat, jää jäljelle mustaa, ja mitä enemmän arvot poikkeavat toisistaan, sitä vaaleammaksi pikseli jää.

Toinen keino kuvan laadun mittaamiseen on SSIM-indeksi, joka tulee sanoista Structural Similarity ja tarkoittaa rakenteellista yhtäläisyyttä. Se on keino mitata havaittavaa laatua digitaalisissa videoissa ja kuvissa, ja sitä käytetään mittaamaan eroa valitun vertailukuvan ja alkuperäislaatuisen lähdekuvan välillä. SSIM-indeksi arvioi kuvasta kolmea komponenttia, luminanssia, kontrastia ja rakennetta, ja niiden muutoksesta johtuvaa visuaalista vaikutusta. Algoritmi toimii siten, että siinä on funktiot signaalin luminanssin vertailulle, kontrastin vertailulle ja rakenteen vertailulle, ja nämä vertailufunktiot yhdistetään yhdeksi funktioksi, joka muodostaa SSIM-indeksin laskentakaavan. SSIM eroaa muista vastaavista tekniikoista siinä, että se arvioi kuvan laadun heikentymistä rakenteellisessa informaatiossa, kun muut vastaavat mittaavat absoluuttista muutosta. Ajatus on, että vierekkäisillä tai samalla alueella olevilla pikseleillä on vahva keskinäinen riippuvuus ja nämä riippuvuudet sisältävät tärkeää tietoa kuvan elementtien rakenteesta. SSIM-indeksi perustuu myös oletukseen, että ihmissilmä on hyvin sopeutunut havaitsemaan rakenteellista informaatiota näkemästään. (52; 53; 54.)

Google on tehnyt kuvaformaatistaan useita erilaisia testejä sen eri ominaisuuksille, ja ne on dokumentoitu yhtiön tuotteistaan kertovilla sivuilla. Google on kerännyt algoritmiavusteisesti verkosta suuren joukon satunnaisia kuvia, joita se on konvertoinut WebP-formaattiin ja verrannut pakkauksen lopputuloksia SSIM-indeksillä. (55.)

Häviöllistä pakkausta varten Google teki kaksi eri testiä neljälle eri kuvakokoelmalle. Kuvakokoelmat sisälsivät sekä tiettyjä valittuja kuvia että noin 11 000 verkosta satunnaisesti haettua kuvaa. Ensimmäisessä testauksessa Google vertasi WebP-formaatin ja JPG-formaatin tiedostokokoa samalla SSIM-indeksillä. Google siis pyrki mittaamaan, kuinka paljon enemmän WebP-formaatilla voi pakata kuvaa, säilyttäen saman laadun. Toisessa testissä Google tarkasteli, kuinka paljon bittejä pikseliä kohden pakkauksen jälkeen on samalla SSIM-indeksillä. (55.)

Tämän työn puitteissa ei lähdetty toistamaan täysin samanlaisia tai yhtä laajoja testauksia, vaan keskityttiin käyttäjän näkökulmaan. Millaisilla pakkausasetuksilla WebP-formaattia kannattaa käyttää ja millaisia silmin havaittavia laatueroja mahdollisesti syntyy, kun kuitenkin pyritään saamaan kuvia pakattua alkuperäistä pienemmäksi verkkosivun latausajan optimoimiseksi. Häviöllisen pakkauksen tapauksessa käytettiin silmämääräisen laadun havaitsemisen lisäksi kuvaan syntyviä eroja esiintuovaa menetelmää, jollainen on esimerkiksi edellä mainittu SSIM-indeksi.

4.1.1 JPG ja häviöllisesti pakattu WebP

JPG- ja WebP-kuvien pakkauksen ja laadun vertailua varten valittiin kuva, joka sisältää erilaisia kuvan pakkauksen kannalta haastavia alueita, kuten kontrasteja, yksityiskohtia ja tasaisia alueita, joissa on vähän yksityiskohtia. Näin se vastaa mahdollisimman monia valokuvatapauksia: kun kuvaan löytyy optimaalinen säätö, se on yleistettävissä useammalle kuvalle.

Testattavaksi valittu valokuva esittää sammakkoa, joka istuu aurinkoisena päivänä puunrungolla. Valittu kuva on kuvassa 7. Kuvan etualan ja taustan välillä on kontrastia, puunrungossa ja kaislikossa on yksityiskohtia ja tausta on tasainen ja siinä on vähän informaatiota. Kuva on melko tyypillinen valokuva.

Kuva 7. JPG-kuvan ja WebP-kuvan vertailua varten valittu kuva.

Kuva on otettu suoraan kamerasta, ja se on kameravalmistajan RAW-muodossa. RAW-muodossa olevia kuvia ei ole prosessoitu lainkaan, vaan ne sisältävät kaiken kameran tallentaman datan alkuperäisessä muodossa eivätkä ne ole sellaisenaan käsiteltävissä kuvankäsittelyohjelmissa. Tästä syystä RAW-kuvat tuli ensin muuntaa bittikarttamuotoon siihen soveltuvalla työkalulla. RAW-formaatit vaihtelevat eri kameravalmistajilla, mutta eri valmistajat yleensä tarjoavat työkalun, jolla muuntaa ne yleisillä kuvankäsittelyohjelmilla käsiteltävään muotoon. (56.)

Valittu kuva on Nikon Electronic Format- eli NEF-formaatissa, joka on Nikon-kameravalmistajan oma RAW-formaatti (57). Kuva on pitkältä sivultaan 3 872 pikseliä ja lyhyeltä sivultaan 2 592 pikseliä. Kuva on resoluutioltaan 240 ppi, mutta sillä ei ole työn kannalta merkitystä luvussa 2.1 todetuista syistä.

RAW-kuva tallennettiin PNG-kuvaksi, jolloin se on pakkaamaton mutta muunnettavissa sekä JPG-formaattiin että häviöllisesti pakattuun WebP-formaattiin. Täysin pakkaamattomalla PNG-kuvalla luodaan mahdollisimman tasa-arvoinen lähtötilanne pakattaville kuville ensimmäistä ja toista testiä varten.

Työn kannalta kuvan ei tarvitse olla niin suuri, kuin se alun perin on, joten se muunnettiin ensin samassa suhteessa pienemmäksi niin, että se on pitkältä sivultaan 1 200 pikseliä ja lyhyeltä sivultaan 803 pikseliä. Kuvan pienentämisessä käytettiin Adobe PhotoShopin Bicubic Sharper -nimistä asetusta, jota on suositellaan käytettäväksi pienentämiseen. Bicubic Sharper käyttää bicubic-interpolaatiota kuvan pienentämiseen ja terävöittää tämän jälkeen kuvaa automaattisesti, koska pienennettäessä kuvan yksityiskohdat saattavat pehmentyä. (58.)

Työssä päätettiin vertailla kuvan laatua käyttämällä SSIM-indeksiä, sillä erot pakatun kuvan ja alkuperäisen kuvan välillä ovat sen verran vähäisiä, että difference-suotimella syntynyt kuva ei antanut riittävän selvää tulosta. SSIM-indeksillä syntyy paitsi vastaava eroavaisuutta esittävä kuva, myös numeraalinen arvo. Tämä numeraalinen arvo on tarkempi vertailuarvo kuin kuva. SSIM-indeksi saatiin laskettua MATLAB-ohjelmalla. MATLAB on MathWorks-yhtiön kehittämä maksullinen ohjelma, jolla voidaan käsitellä, analysoida ja visualisoida dataa (59).

Vertailu tehdään määrittämällä ensin referenssikuva eli alkuperäinen kuva, johon pakattua kuvaa verrataan. Lisäksi määritellään kuva, jota verrataan referenssikuvaan. Esimerkkikoodissa 4 on esimerkki koodista, jolla voidaan verrata JPG-kuvan pakkausta alkuperäiseen PNG-kuvaan. Koodi tulostaa erotuskuvan ja numeerisen arvon.

```
referenssikuva = imread('valittukuva.png');
vertailukuva = imread('valittukuva.jpg');
[ssimval, ssimmap] = ssim(vertailukuva, referens-
sikuva );
figure, imshow(ssimmap,[]);
title(ssimval);
```

Esimerkkikoodi 4. MATLAB-koodi JPG-kuvan ja PNG-kuvan vertailuun.

Työn yhteydessä kuvalle toteutettiin kolme erilaista testiä kuvan pakkausta mitattaessa. Kaikkia testejä varten haettiin ensin kuvalle optimaalinen pakkaus JPG-formaatissa. Tavoitteena oli, että kuva-artefakteja on mahdollisimman vähän havaittavissa, mutta kuva on kuitenkin mahdollisimman pakattu suhteessa laatuun.

Tallentamalla PNG-kuva JPG-kuvaksi eri pakkauskertoimilla valikoitui lopullisen kuvan pakkauskertoimeksi 8, asteikolta 1–12. Tallennusvalikosta valittiin asetus Baseline Optimized, joka pakkaa syntyvän kuvatiedoston hieman pienemmäksi ja optimoi värejä (60). Tämä pakkausasetus antoi hyväksyttävän laadun silmämääräisesti havaittuna ja SSIM-indeksillä tulokseksi 0,9742. Syntyneen kuvatiedoston koko on 209 KiB.

Ensimmäisessä testissä PNG-kuva pakattiin WebP-formaattiin tavoitellen samaa kuvanlaatua kuin JPG-kuvassa on. Tarkoitus oli nähdä, miten paljon WebP-kuvan tiedostokoko eroaa samanlaatuisen JPG-kuvan tiedostokoosta. Kuvanlaatua tarkasteltiin silmämääräisesti rinnakkain vertaillen sekä SSIM-indeksillä. Kuva konvertoidaan WebP-formaattiin komentotulkissa. Halutun pakkauskertoimen voi valita asteikolta 0–100, ja siihen on mahdollista valita erilaisia asetuksia, kuten lähdekuvan tyyppi. Toinen huomionarvoinen asetus on metadata. Tässä työssä valittu testikuva oli sRGB-väriprofiilissa, joka on sopivin väriprofiili näytöillä katsomiseen. Mikäli kuva on adobeRGB-väriprofiilissa, joka on myös yleinen väriprofiili mutta sopivampi tulostamiseen, täytyy konvertoinnin yhteydessä asettaa asetus -metadata all. Metadata tuo mukanaan kuvan sisältämän väriprofiilin, mutta lisää kuvatiedoston kokoa. Poikkeava väriprofiili tulee ottaa metatiedoissa mukaan, jos ei halua kuvan värien muuttuvan pakkauksen yhteydessä. WebP käyttää sRGB-väriprofiilia, joten metadataa ei tarvitse huomioida, jos lähdekuva on tässä väriprofiilissa. (61.)

Usean kokeilun jälkeen pakkauskertoimeksi valittiin 88 ja lähdekuvan tyypiksi valittiin photo eli valokuva pakattavan kuvan luonteen vuoksi. Valituilla asetuksilla konvertointikomento on cwebp -preset photo -q 88 valittukuva.png -o valittukuva.webp. Näillä asetuksilla päästiin lähimmäksi pakatun JPG-kuvan laatua sekä silmämääräisessä tarkastelussa että SSIM-indeksillä mitattuna. SSIM-indeksillä tulos on 0,9751 ja syntyneen tiedoston koko 146 KiB. Lähes samalla SSIM-indeksi arvolla WebP-tiedoston koko on noin 30 % pienempi kuin verrattavan JPG-kuvan. SSIM-erotuskuvia rinnakkain vertaillessa voi havaita, että JPG-pakkauksessa yksityiskohtiin on tullut voimakkaammin muutosta kuvan etualalla olevan puukappaleen yksityiskohtiin, kun WebP-pakkauksessa yksityiskohtiin on tullut vähemmän muutosta mutta hieman laajemmin vastaavalla alueella. SSIM-erotuskuvat on esitetty kuvassa 8.

Kuva 8. Vasemmalla JPG-pakkauksen SSIM-erotuskuva ja oikealla WebP-pakkauksen SSIM-erotuskuva.

Toisessa testissä lähestyttiin pakkausta kuvanlaadun sijasta kuvatiedoston koon kautta. WebP-kuva pakattiin toisessa testissä niin, että sen tiedostokoko vastaa mahdollisimman lähelle JPG-kuvan tiedostokokoa. WebP-formaattiin pakattaessa on mahdollista asetuksella asettaa tavoitekoko syntyvälle tiedostolle, mitä hyödynnetään tässä. Asetus tapahtuu -size komennolla, jonka perään kirjoitetaan haluttu tavoitekoko tavuina. Tämän jälkeen voidaan analysoida, millainen laatuero kuvissa on alkuperäiseen PNG-kuvaan verrattuna, ja verrata tuloksia. Koska asetuksella voi asettaa vain tavoitteen syntyvän tiedoston koolle, ei WebP-kuvaa ole mahdollista pakata täydellisesti samankokoiseksi kuin JPG-kuvaa. Tästä syystä WebP-kuva pakataan hieman pienemmäksi, jotta ei synny laatua vertailtaessa etua tutkittavan kohteen puolelle verrattuna lähteeseen.

Valituilla asetuksilla komento on cwebp -preset photo -size 214 000 valittukuva.png -o valittukuva.webp, ja tällä asetuksella pakatun WebP-tiedoston koko on 203 KiB. Silmämääräisesti laatueroa ei ole juurikaan havaittavissa, sillä suurin osa molemmissa formaateissa tapahtuvasta pakkauksesta ilmenee etualan puunrungossa, josta ihmissilmän on hankala havaita muutosta pienten yksityiskohtien runsaan määrän vuoksi. SSIM-indeksillä ero on kuitenkin huomattavissa. Näillä asetuksilla pakatun WebP-kuvan tulos on 0,9822 ja erotuskuvassa ei ole nähtävissä juurikaan eroa PNG-kuvaan. Mielenkiintoista on, että eroa ei ole juuri lainkaan alueella, jossa JPG-pakatussa kuvassa eroa on eniten. Samalla tiedostokoolla WebP-kuvan laatu on parempi kuin JPG-kuvan.

Yleinen ja todennäköinen tilanne on, että pyrkimyksenä on saada olemassa oleva JPG-kuva pakattua pienemmäksi tinkimättä kuvan laadusta. Tällöin on mahdollista, että alkuperäistä pakkaamatonta kuvaa ei ole saatavilla eli kertaalleen pakattu JPG-kuva täytyy pakata pienemmäksi. Tällainen esimerkkitapaus vastaa sitä, millaiseen käyttöön WebP tällä hetkellä verkossa parhaiten soveltuu. WebP-formaatilla ei ole tällä hetkellä mahdollista täysin korvata JPG-formaattia sen rajallisen tuen vuoksi, joten WebP-kuvia voidaan käyttää tarjoamaan pienempi vaihtoehto JPG-kuvalle silloin, kun selaimessa on tuki.

Kolmannessa testissä pakattiin PNG-kuvasta tallennettu JPG-kuva WebP-formaattiin, jotta voitiin havaita, kuinka paljon lisää WebP pystyy pakkamaan jo pakattua JPG-kuvaa säilyttäen mahdollisimman hyvin laadun. Asetuksina käytettiin ensimmäisen testin asetuksia, joilla saatiin tiedostokokoa pienennettyä mutta säilytettyä hyvä laatu. Vertailtaessa JPG-kuvasta pakattua WebP-kuvaa ei silmin ole havaittavissa eroa. Kun näiden kahden kuvan eroa tarkastellaan SSIM-indeksillä, on tulos 0,9819 ja erotuskuvassa on havaittavissa pientä eroa etualan puunrungon yksityiskohtien pakkauksessa. Kun syntynyttä WebP-kuvaa verrataan alkuperäiseen pakkaamattomaan PNG-kuvaan, on eroa selkeämmin. Silmin eroa on edelleen vaikea havaita, mutta SSIM-erotuskuvassa näkyy sekä JPG-pakkauksen synnyttämä ero, että siihen lisätty WebP-pakkauksen synnyttämä ero, ja tulos on 0,9653. Syntyneen tiedoston koko on 142 KiB, eli tiedosto on noin 32 % pienempi kuin JPG-tiedosto.

WebP-kuvalla on vähäisissä pakkausmäärissä hyvät tulokset, vielä 75 pakkauskertoimellakin, joka on oletuspakkauskerroin (62). WebP-pakkauksen laatu laskee kuitenkin todella nopeasti suuremmalla pakkausmäärällä, ja 50-pakkauskertoiminen WebP-kuva on jo selvästi huonolaatuisempi, kuten esimerkiksi kuvassa 9. Suuremmilla pakkausmäärillä havainnollistuu hyvin JPG-pakkauksen ja WebP-pakkauksen ero. JPG-pakkauksessa tulee hyvin ilmi algoritmin lohkopakkausmenetelmä, kun kuvan artefaktit esiintyvät lohkomaisesti. WebP-pakkaus puolestaan hukkaa kuvan yksityiskohtia algoritmin aiheuttamalla sumennuksella.

Kuva 9. Vasemmalla JPG-kuva ja oikealla WebP-kuva rinnakkain vertailussa.

JPG-pakkaus pärjää WebP-pakkausta paremmin suurilla pakkausmäärillä. Kuvan 9 rinnakkainvertailussa käy ilmi JPG-kuvan parempi kyky säilyttää yksityiskohdat suurillakin pakkausmäärillä, mutta WebP-kuvassa ilmenee vähemmän selkeitä artefakteja kuin JPG-kuvissa.

4.1.2 PNG ja häviöttömästi pakattu WebP

PNG-formaattia ja WebP-formaattia vertailtiin PNG-kuvalla, joka on ladattu testiä varten ilmaisia PNG-kuvia tarjoavalta sivulta. Valitun PNG-kuvan bittisyvyys on 24, ja sen tiedostokoko on 92,8 KiB. Kuvassa 10 on valittu PNG-kuva, joka esittää piirakkadiagrammia. Valitulla PNG-kuvalla on erilaisia pakkauksen kannalta mielenkiintoisia ominaisuuksia, kuten läpinäkyvä tausta, useita värejä, tarkkoja diagonaaleja reunoja ja liukuväri. Vertailuja tehtiin kaksi: ensin niin, että kuvalla ei ole taustaa, eli alpha-kanava on käytössä, ja sen jälkeen niin, että kuvalla on valkoinen tausta ja värejä on karsittu, jolloin myös verrattava PNG-kuva on pienempi.

Kuva 10. Valittu PNG-testikuva.

PNG-kuva pakataan WebP-formaattiin samalla komennolla kuin JPG-kuva, eli cwebp. Lisäksi asetuksiin asetetaan -lossless, jotta kuva pakataan häviöttömästi. Tässä oletusasetus on pakkauskertoimella 75, mutta sen voi halutessaan asettaa pienemmäksi tai suuremmaksi. Pienempi luku mahdollistaa nopeamman pakkaamisen mutta suuremman tiedostokoon, ja suurempi luku päinvastoin aiheuttaa hitaamman pakkauksen mutta syntyvän tiedoston koko on pienempi. Alphakanavalle on myös mahdollista asettaa pakkauskerroin komennolla -alpha_q, jolloin 100 on häviötön ja kaikki alle 100 on häviöllistä. (62.)

Taulukossa 1 on esitetty tulokset syntyneiden WebP-kuvien koosta verrattuna alkuperäiseen PNG-kuvaan. Häviötön pakkaus -asetuksella kuvanlaatu ei muuttunut silmin nähden lainkaan, kuten ei pidäkään. Tiedoston koko pieneni huomattavasti kaikilla käytetyillä pakkauskertoimilla 0, 75 ja 100. Komentotulkissa komento pakkauskertoimella 100 on cwebp -lossless -q 100 esimerkki.png -o esimerkki.webp. Suurimmalla mahdollisella pakkauksella tiedoston koko pieneni 39 %. Tämä on suurin piirtein linjassa Googlen omien testien tuloksien kanssa.

Taulukko 1. 24-bittisen PNG-kuvan ja häviöttömästi pakattujen WebP-kuvien vertailu.

Alkuperäinen PNG-tiedosto		92,8 KiB
WebP oletusasetuksilla	cwebp -lossless esimerkki.png -o esimerkki.webp	57,7 KiB
WebP pakkauskertoimella 0	cwebp -lossless -q 0 esimerkki.png -o esimerkki.webp	68,8 KiB
WebP pakkauskertoimella 100	cwebp -lossless -q 100 esimerkki.png -o esimerkki.webp	56,6 KiB

Toista testiä varten sama kuva tallennettiin 8-bittisenä PNG-kuvana. Kuva tallennettiin myös valkoisella taustalla, jolloin alpha-kanavaa ei tarvitse huomioida. Näillä asetuksilla tallennetun PNG-kuvan koko on 31,5 KiB. Tulokset samoilla WebP-asetuksilla kuin edellisessä testissä olivat samansuuntaiset, mutta pakkaus on vähäisempää, kuten on esitetty taulukossa 2. Suurimmalla WebP-pakkauksella kuva pieneni noin 21,9 % PNG-kuvasta. Pakkaus ei ollut yhtä tehokas kuin ensimmäisessä testissä, mutta tämä on ymmärrettävää, sillä pakattava kuva on lähtökohtaisesti pienempi tiedostokooltaan.

Taulukko 2. 8-bittisen PNG-kuvan ja häviöttömästi pakattujen WebP-kuvien vertailu.

Alkuperäinen PNG-tiedosto		31,5 KiB
WebP oletusasetuksilla	cwebp -lossless esimerkki.png -o esimerkki.webp	24,8 KiB
WebP pakkauskertoimella 0	cwebp -lossless -q 0 esimerkki.png -o esimerkki.webp	26,9 KiB
WebP pakkauskertoimella 100	cwebp -lossless -q 100 esimerkki.png -o esimerkki.webp	24,6 KiB

4.1.3 GIF ja animoitu WebP

GIF-formaatin ja WebP-formaatin vertailuun valittiin GIF-animaatioita sosiaaliseen mediaan jaettavaksi tarjoavalta sivulta yksi GIF-animaatio. Valittu GIF-animaatio on kestoltaan noin puolitoista sekuntia ja hyvälaatuinen siten, että siinä ei ole helposti silmin havaittavia artefakteja. GIF-tiedoston koko on 998 KiB. Testissä tarkasteltiin useita eri asetuksia käyttämällä, kuinka paljon lisää GIF-tiedostoa voidaan WebP-formaattiin konvertoimalla pakata lisää.

Oletusasetuksilla konvertointi on yksinkertaisin tapa muuntaa GIF-animaatio WebP-animaatioksi, ja sen komentoriviin kirjoitettava komento on gif2webp esimerkki.gif -o esimerkki.webp. Gif2webp on käytettävä konvertteri, eli se muuntaa tiedoston GIF-formaatista WebP-formaattiin. Esimerkki.gif on tiedosto, joka halutaan konvertoida. -o ja sitä seuraava esimerkki.webp kertovat, että konvertoinnissa halutaan luoda uusi esimerkki.webp-niminen tiedosto. Mikäli muita valintoja ei ole annettu, konvertointi tehdään oletuksena pakkaamattomana ja 75-pakkauskertoimella. (63.)

Konvertointia tehtäessä komentorivin komentoon on mahdollista antaa asetuksia, kuten pakataanko tiedosto häviöttömästi, häviöllisesti vai molemmilla tavoilla, sekä mitä pakkauskerrointa käytetään tai kopioidaanko mahdollisia metatietoja. Googlen ohjeistuksessa on lueteltu gif2webp-konvertterissa mahdolliset asetukset ja mitä ne ovat oletusarvoltaan, mikäli niitä ei erikseen määritellä. (63.)

Häviöttömässä konvertoinnissa pakkauskerroin voi olla välillä 0–100, ja tämä ilmoitetaan komentorivin komennossa kirjoittamalla konvertterin perään -q ja haluttu luku, esimerkiksi 100. Mitä pienempi pakkauskerroin on, sitä nopeammin konvertointi tapahtuu mutta sitä suurempi luotu tiedosto on. Isommalla pakkauskertoimella konvertointi on vastaavasti hitaampi, mutta syntyvä tiedosto pienempi. (63.)

Häviötön pakkaus tehtiin kolmella eri asetuksella valitulle GIF-animaatiolle. Tulokset eri asetuksilla konvertoinnista on esitetty taulukossa 3. Ensimmäinen testi tehtiin oletusasetuksilla eli 75-pakkauskertoimella. Toinen testi tehtiin 0-pakkauskertoimella ja kolmas 100-pakkauskertoimella. Koska kyse on häviöttömästi pakkaavasta asetuksesta, syntynyt WebP-tiedosto ei eronnut laadullisesti alkuperäisestä GIF-tiedostosta, se oli vain pakattu pienemmäksi.

Taulukko 3. GIF-tiedoston ja siitä konvertoitujen WebP-tiedostojen koot.

Alkuperäinen GIF-tiedosto		998 KiB
WebP oletusasetuksilla	gif2webp esimerkki.gif -o esimerkki.webp	808 KiB
WebP pakkauskertoimella 0	gif2webp -q 0 esimerkki.gif -o esimerkki.webp	842 KiB
WebP pakkauskertoimella 100	gif2webp -q 100 esimerkki.gif -o esimerkki.webp	806 KiB

Tämän jälkeen tehtiin vastaavat konvertoinnit myös häviöllisellä pakkauksella ja asetuksella, joka hyödyntää sekä häviöllistä että häviötöntä pakkausta optimaalisen lopputuloksen saavuttamiseksi. Konvertteri on mahdollista asettaa käyttämään häviöllistä pakkausta lisäämällä käskyyn -lossy. Asetuksella -mixed konvertteri optimoi pakkauksen valitsemalla kuvakehys kerrallaan, käyttääkö se häviöllistä vai häviötöntä pakkausta. Häviöllisen pakkauksen tapauksessa aiempi laatuasetus -q toimii siten, että pienempi luku johtaa pienempään syntyvään tiedostoon mutta alhaisemmalla laadulla, ja isompi luku vastaavasti paremmalla laadulla mutta suuremmalla tiedostokoolla. Edellä kuvatut testit tehtiin uudestaan muuten samoilla asetuksilla, mutta häviöllisenä pakkauksena ja vielä yksi erillinen testi -mixed-asetuksella. (63.) Tulos on nähtävissä taulukossa 4.

Taulukko 4. GIF-tiedoston ja siitä häviöllisesti konvertoitujen WebP-tiedostojen koot.

Alkuperäinen GIF-tiedosto		998 KiB
WebP oletusasetuksilla	gif2webp -lossy esimerkki.gif -o esimerkki.webp	155 KiB
WebP pakkauskertoimella 0	gif2webp -lossy -q 0 esimerkki.gif -o esimerkki.webp	759 KiB
WebP pakkauskertoimella 100	gif2webp -lossy -q 100 esimerkki.gif -o esimerkki.webp	17,3 KiB
WebP mixed asetuksella	gif2webp -mixed esimerkki.gif -o esimerkki.webp	155 KiB

Erot ovat jo huomattavasti suurempia kuin häviöttömässä pakkauksessa, niin tiedostokoon kuin kuvanlaadun suhteen, kuten näkyy kuvassa 11.

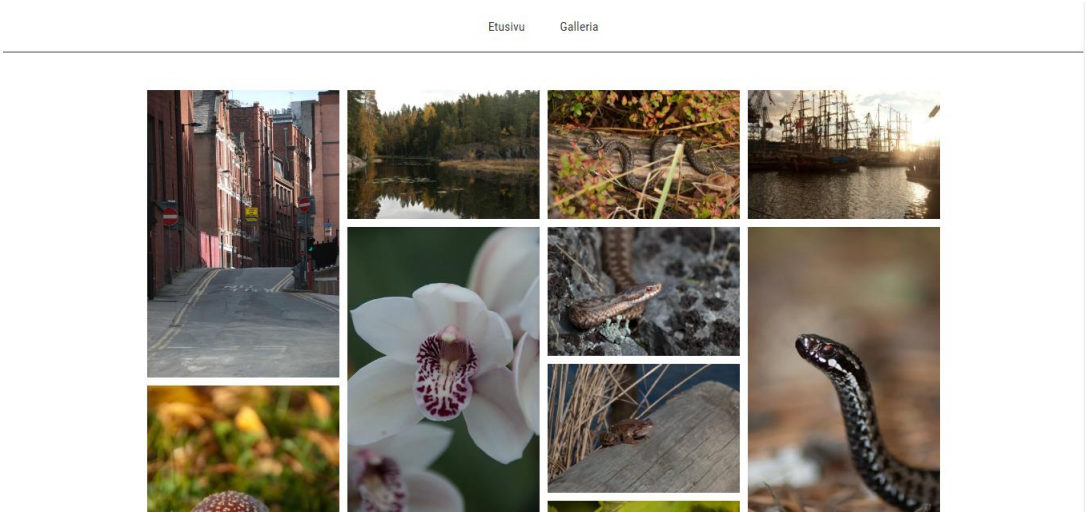
Kuva 11. Kuvan laadun ero häviöllisesti pakatussa WebP-formaatissa. Vasemmalla alkuperäinen GIF-animaatio, keskellä 75-pakkauskertoimella pakattu WebP-animaatio ja oikealla 0-pakkauskertoimella pakattu WebP-animaatio.

4.2 Testiympäristö

Testiympäristössä verrattiin vain WebP-kuvien ja JPG-kuvien vaikutusta sivunlatausnopeuteen. Valinta perustuu oletukseen, että usein verkkokuvagalleriat esittelevät valokuvia, joiden optimaalisin verkkokuvaformaatti on JPG. GIF on PNG-formaatin yleistyttyä harvemmin esiintyvä formaatti, ellei kyseessä ole animaatio, joten sen vaikutuksen ei nähty olevan tarpeeksi suuri vertailuun tässä tapauksessa. PNG puolestaan soveltuu parhaiten selkeälle grafiikalle, jossa on voimakkaita kontrasteja ja vähemmän värejä. Ne ovat yleisesti sivuilla käytössä logoissa tai informatiivisissa kuvakkeissa ja yksittäisissä kuvituksissa, joten niidenkään koon vaikutusta ei ole nähty tässä tapauksessa yhtä suurena kuin täysväristen JPG-kuvien.

Testi toteutettiin kaksi kertaa. Ensimmäisellä kerralla testissä verrattiin WebP-kuvien ja JPG-kuvien latausnopeutta kymmenen kuvan kanssa ja toisella kerralla 50 kuvan kanssa. Tavoitteena oli ensimmäisellä kymmenellä kuvalla tehdä tarkempi vertaus kuvaformaattien vaikutuksesta sivunlatausnopeuteen ja 50 kuvan erällä tarkastella kuinka paljon moninkertainen määrä kasvattaa eroa.

Työssä käytettävä galleria on rakenteeltaan yksinkertainen HTML-verkkosivu, joka sisältää kaksi erillistä sivua. Yhdellä sivulla on tekstiä esittelemään sivun tarkoitus, ja toisella erillisellä sivulla ovat varsinaisen gallerian kuvat. Kuvassa 12 on ruutukaappaus toteutetusta galleriasivusta. Koska työssä on tarkoituksena mitata WebP-kuvien pakkauksen ja WebP-kuvien palvelimelta lataamisen vaikutusta sivunlatausnopeuteen, ladataan kuvat sivulle kaikki kerralla. Sivun optimoinnin kannalta kuvat olisi parempi ladata vaiheittain, esimerkiksi tuottamalla erikseen pienempikokoiset esikatselukuvat ja lataamaan isot kuvat vasta, kun ne on valittu sivulla katsottavaksi, jolloin ne eivät kuormita sivua ladattaessa yhtä paljon. Tässä testissä kuvat ladataan kuitenkin kerralla täysikokoisina, jotta saadaan korostettua kuvan pakkauksen vaikutusta. Erillisiä pienempiä esittelykuvia ei siis tarjota ensilatauksen yhteydessä, mutta ne asetetaan kuitenkin visuaalisuuden vuoksi CSS-tyyleillä näkymään pienempinä. Kuvat ladataan ruudukkoon, ja kuvaa klikkaamalla se aukeaa suuremmaksi. Dynaamisen ruudukon muodostamiseen käytetään erillistä JavaScript-kirjastoa, mutta sillä ei ole työhön muuta vaikutusta kuin tarjota visuaalisesti miellyttävä ja responsiivinen tapa asetella kuvat. Testiympäristö on muuten optimoitu Verkkosivun optimointi -luvun mukaisesti ja pyritty toteuttamaan oikean gallerian kaltaiseksi.



Kuva 12. Toteutettu galleriasivu.

Galleriaa testattiin kahdella palvelulla. Ensimmäinen palvelu on Googlen PageSpeed Insights. Se testaa kuvaa sekä mobiiliversiona että työpöytäversiona. PageSpeed Insights antaa tuloksen, kuinka monta pistettä sadasta sivu saa kummallakin versiolla sekä onko saatu tulos punainen, keltainen vai vihreä. Punainen tarkoittaa, että tulos on huono, keltainen että parannettavaa on ja vihreä hyvää. Tämän lisäksi PageSpeed

Insights listaa käytetyt testikriteerit ja ilmoittaa ovatko ne täyttyneet hyvin, onko niihin suositeltavia parannusehdotuksia ja ovatko ne tärkeitä parannuksia. (64.)

Toinen käytettävä palvelu on avoimen lähdekoodin palvelu YSlow. YSlow käyttää Yahoo-yrityksen suorituskykyyn erikoistuneen tiimin tunnistamia sääntöjä, jotka vaikuttavat verkkosivun suorituskykyyn. YSlow kertoo sivun tuloksen asteikolta A–F, jossa A on paras ja F huonoin. YSlow niin ikään kertoo tulokset väreillä, joita on kuitenkin enemmän kuin PageSpeed Insights -palvelulla. YSlow-palvelun käyttämät värit ovat punainen, oranssi, keltainen, vaalean vihreä ja tumman virheä, alkaen huonosta ja päättyen hyvään. (65.)

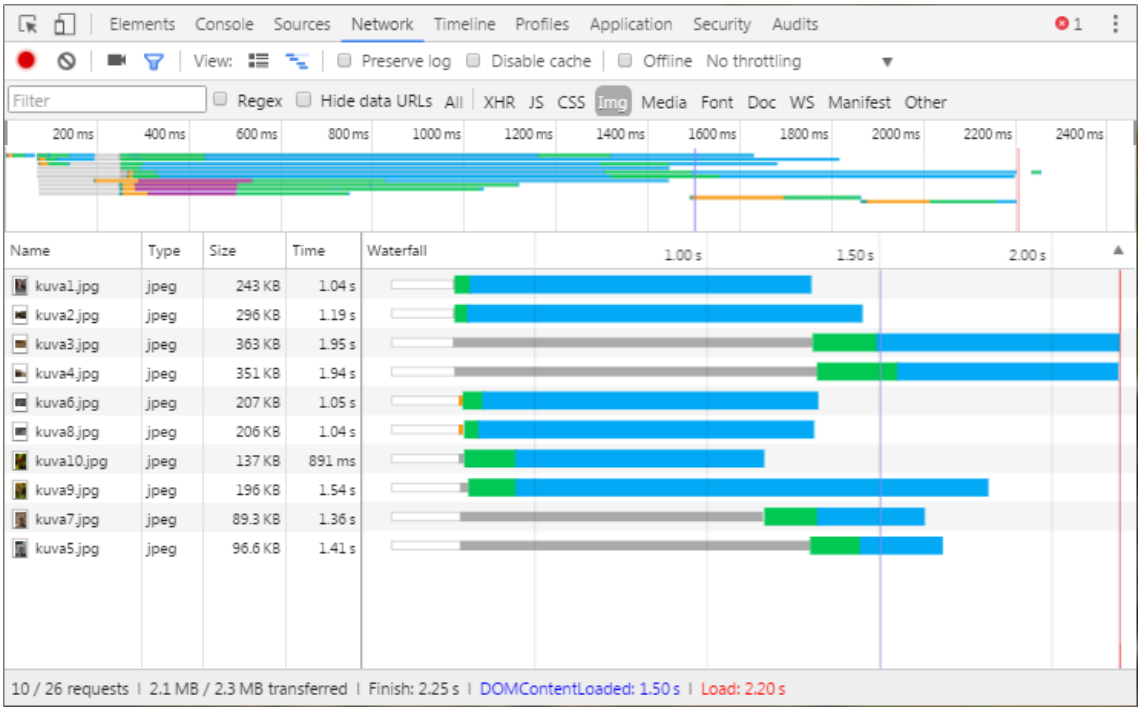
4.3 Tulokset

Luvuissa 4.1.1–4.1.3 tehtyjen testien osalta voidaan todeta, että WebP-formaatti pakkaa kuvia selvästi pienemmäksi vaikuttamatta merkittävästi laatuun. PNG- ja GIF-kuvien osalta kyseessä on häviötön pakkaus, ja niissä ei tästä syystä ollut vaikutusta kuvan laatuun. Tiedostokokojen havaittiin olevan noin 30 % pienempiä kuin alkuperäisten, mikä on linjassa Googlen tekemien havaintojen kanssa. Myös häviöllisen pakkauksen osalta voidaan todeta, että WebP-formaattiin pakattu kuvatiedosto on pienempi kuin JPG-kuva. Testeissä WebP-formaatti pakkasi kuvat pienemmiksi säilyttäen paremman laadun tai vastaavan laadun. Myös jo pakatun JPG-kuvan edelleen pakkaaminen WebP-formaattiin tuotti pienemmän tiedostokoon vähäisellä vaikutuksella kuvan laatuun.

Kun kuvia tarkastellaan verkkosivulla, JPG-kuvia sisältävä galleria saa huonot tulokset Googlen PageSpeed Insights -palvelun testissä. Sivun työpöytäversio saa tuloksen 33/100 ja mobiiliversio 37/100, eli molemmat ovat punaisella. Sivujen ensimmäinen parannusehdotus on kuvien optimointi: palvelun mukaan kuvien kokoa on mahdollista optimoida 73–91 %. Toissijaisia parannusehdotuksia ovat muut tyypilliset optimointikohteet, kuten selaimen välimuistin hyödyntäminen, sisällön latausjärjestyksen parantaminen ja tiedostojen pakkauksen mahdollistaminen.

YSlow-palvelulla testattaessa JPG-kuvia sisältävä galleria saa paremman tuloksen. Mittarina käytettiin pienelle sivulle tehtävää testiä, jolloin palvelinpuolen parannuskriteerejä on vähemmän. Tulos on B ja värikoodina on vaaleanvihreä. Parannuskehotukset

ovat samanlaiset kuin PageSpeed Insights -palvelun tuloksessa, kuten välimuistin hyödyntäminen ja tiedostojen pakkauksen mahdollistaminen. Lisäksi on muutama muu parannusehdotus, joita PageSpeed Insights -tuloksissa ei ollut, mutta niillä ei ole tämän työn kannalta merkitystä. Kuvien suhteen YSlow-palvelulla on parannuskehotus, mutta vain osittain samasta syystä kuin PageSpeed Insights -palvelussa. Siinä missä PageSpeed Insights kehottaa sekä pakkaamaan kuvia että muuttamaan niiden kokoa, YSlow kehottaa olemaan pienentämättä kuvia HTML-koodissa. Kuvien pitäisi olla sen kokoisia, kuin minkäkokoisina ne esitetään sivulla. Kuvan pakkaukseen YSlow ei ota kantaa. Kuvien koko on yhteensä 2 233,7 tavua ja kaikkien sivun elementtien yhteenlaskettu koko on 2 425,8 tavua. Kuvassa 13 on ruutukaappaus gallerian latausnopeuksista Chrome Developer Tools -työkalulla katsottuna.

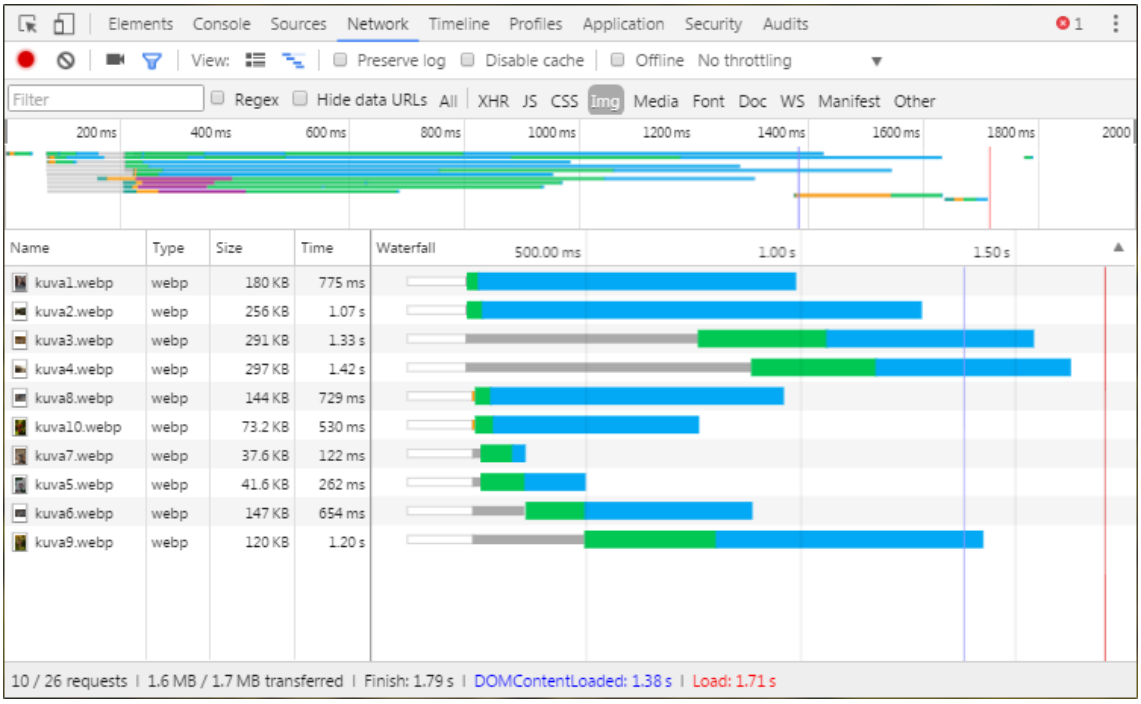


Kuva 13. Gallerian lataustiedot katsottuna Chrome Developer Tools -työkalulla, kun kuvasisältö on JPG-kuvia.

WebP-kuvia sisältävä galleria saa huomattavasti paremmat tulokset PageSpeed Insights -testissä. Työpöytäversion tulos on 86/100 ja väri on vihreä. Kuvien optimoinnista ei ole mitään mainintaa, vaan se on siirtynyt hyväksyttyjen sääntöjen listalle. Siitä huolimatta, että vain kompressointikehotus toteutui, kuvat ovat pikselimitoiltaan edelleen yhtä suuret kuin JPG-kuvat. Mitään tärkeää parannuskehotusta ei tämän jälkeen ole, mutta harkittavien parannusten listalla on muuten samat parannusehdotukset kuin JPG-kuvillakin. Mobiiliversion tulos on 79/100 ja keltainen. Mobiiliversiossa välimuistin

hyödyntäminen on punaisella listalla eli tärkeä parannus. Muuten tulos on sama kuin työpöytäversiossa.

YSlow-palvelun tulos WebP-kuville on sama kuin JPG-kuville. Tämä on odotettavaa, sillä YSlow ei ota kantaa kuvien kompressointiin, vain niiden esityksen kokoon, joka ei eroa valituissa WebP-kuvissa ja JPG-kuvissa. Kuvien koko on yhteensä 1 622,6 tavua ja kaikkien sivun elementtien yhteenlaskettu koko on 1 814,7 tavua. Kuvassa 14 on ruutukaappaus gallerian latausnopeuksista Chrome Developer Tools -työkalulla katsottuna.



Kuva 14. Gallerian lataustiedot katsottuna Chrome Developer Tools -työkalulla, kun kuvasisältö on WebP-kuvia.

Vertaamalla JPG-kuvia sisältävän gallerian ja WebP-kuvia sisältävän gallerian lataustietoja Chrome Developer Tools -työkalun Network-välilehdellä voi tarkastella eri latausaikojen kestoja. Ne ovat joka kerta hieman erilaisia riippuen yhteydestä ja palvelimen vastausajasta, mutta lataamalla sivut useamman kerran saatiin keskimääräinen tulos. Molemmat galleriat ladattiin kymmenen kertaa, ja näiden latauskertojen keskiarvosta on havaittavissa, että WebP-kuvia sisältävä galleria latautuu keskimäärin 1,576 sekunnissa ja JPG-kuvia sisältävä galleria keskimäärin 2,056 sekunnissa. WebP-

galleria latautuu siis keskimäärin 0,48 sekuntia nopeammin silloin, kun kuvat eivät tule selaimen välimuistista vaan ladataan palvelimelta. Prosentteina eroa on noin 23,35 %.

Toistettuna 50 kuvalla tulokset vastaavat kymmenen kuvan tulosta. WebP-kuvilla galleria latautuu keskimäärin 6,888 sekunnissa ja JPG-kuvilla 9,374 sekunnissa. Eroa sivujen latauksessa on keskimäärin 2,486 sekuntia, eli noin 26,5 %, mikä on vähän enemmän kuin kymmenellä kuvalla.

Kymmenellä kuvalla toteutetun testin puoli sekuntia ei ole vielä aikaero, jonka käyttäjä helposti huomaa, mutta 50 kuvalla toistetun testin 2,5 sekuntia on jo huomattava ero latausajassa. Kuvien tehokkaampi pakkaus on aidosti huomionarvoinen asia, jos kuvia on sivulla paljon. Huomioitavaa on, että selaimet usein tallentavat kuvat välimuistiin, joten tulevilla käynneillä sivulle kuvat latautuvat nopeammin, jos välimuistia ei ole välissä tyhjennetty. Ensimmäinen latauskerta voi kuitenkin olla kriittinen käyttäjän ensivaikutelman kannalta, joten sen on tärkeää olla optimaalinen. Mobiiliselaimella latausajat ovat kriittisempiä kuin työaseman selaimella, mutta 26 %:n ero latausajoissa varsinkin hitaammilla yhteyksillä on merkittävä ero kummallakin alustalla.

5 Yhteenveto

Insinöörityön ensimmäisessä osassa perehdyttiin digitaalisiin kuviin ja käytiin läpi, mitä ne ovat ja mitä ominaisuuksia niillä on. Tämän jälkeen tutustuttiin tarkemmin, millä eri tavoin niitä voidaan pakata ja miten ne eroavat toisistaan. Lisäksi käytiin läpi kolme yleisintä verkossa tänä päivänä käytettyä kuvaformaattia, joita käytettiin myöhemmin työssä vertailuun WebP-formaattia vasten. Työssä käsiteltiin lyhyesti myös kuvien optimointia verkkosivun sisällön kannalta. Toisessa osassa perehdyttiin WebP-formaattiin, käsiteltiin mikä se on, miten sen pakkaus toimii ja miten sitä käytetään. Tämän lisäksi tutkittiin sen nykyistä tukea ja pohdittiin sen mahdollisia tulevaisuuden näkymiä viimeisimpien sitä koskevien uutisten valossa.

Varsinaisessa työosuudessa verrattiin WebP-formaattia sen eri ominaisuuksien osalta aiemmin työssä esiteltyihin formaatteihin. Näissä vertailuissa pidettiin mielessä Googlen omat vertailut ja todettiin, että samassa laajuudessa ei vertailuja tulla toteuttamaan. Vertailussa valittiin kullekin kuvaformaatille tyypillinen kuva. Häviöllisen pakkauksen osalta pidettiin huolta tasapuolisesta vertailusta tekemällä molempien formaattien osalta pakkaus alun perin pakkaamattomaan kuvaan. Lisäksi tehtiin WebP-formaattiin pakkaus jo pakatulle kuvalle, jotta saatiin selville, kuinka paljon lisää se voi ennestään tiivistää pakattua kuvaa. Häviöttömien formaattien osalta valitut kuvat pakattiin uudelleen käyttäen WebP-formaatin häviötöntä ominaisuutta ja animaatio-ominaisuutta. Ensimmäisessä työosuudessa saatua tulosta hyödynnettiin työn toisessa työosuudessa. Siinä verrattiin kuvagallerian kuvien latausta eri kuvamäärillä, kun sisältönä oli ensimmäisellä kerralla JPG-kuvia ja toisella kerralla samat kuvat pakattuna luvussa 4.1.1 määritellyin asetuksin WebP-formaattiin.

Työssä lähdettiin hakemaan varmennusta Googlen testituloksille ja tarkastelemaan sen hyötyä oikeassa ympäristössä. Työn tavoitteessa onnistuttiin: saatiin toistettua Googlen tuloksia vastaavat tulokset perustelluin menetelmin. Lisäksi voitiin konkreettisesti todeta hyöty sivunlatausajassa, kun sisältö on optimoitu pakkaamalla kuvat pienemmiksi ilman, että kuvan laadusta on tarvinnut tinkiä.

Työn aikana tuli kuitenkin todettua, että puolueettomuus työkalujen valinnassa on tärkeää. Kun sivuja tutkittiin Googlen omalla työkalulla, PageSpeed Insightsilla, olivat tulokset liiankin hyvät. Pelkästään kuvien pakkaaminen Googlen omaan formaattiin riitti poistamaan muitakin sivun kuviin liittyviä ongelmia. Toinen optimointityökalu YSlow ei

ottanut kantaa kuvien pakkaukseen eikä sillä ilmennyt mitään eroa eri formaattien suhteen.

Lisäksi formaatin valinnan kannalta on pidettävä mielessä sen toistaiseksi vajaa tuki ja hankala käytettävyys. WebP ei ole yhtä laajalti tuettu kuin kuvaformaatit, joita sen sanotaan haastavan, joten sitä ei voi ottaa korvaavana kuvaformaattina käyttöön. Jääkin käyttäjän päätettäväksi, onko useamman kuin yhden kuvaformaatin käyttäminen järkevää optimointihyötyihin nähden. WebP-kuvien käsittely ei onnistu käyttöjärjestelmillä suoraan, mikä myös heikentää sen käytettävyyttä. Työssä esitellyt JPEG2000 ja JPG-XR ovat myös tehokkaita pakkaustyökaluja, jotka säilyttävät kuvan laadun hyvin, mutta ne eivät rajallisen tukensa vuoksi ole nousseet suureen suosioon, vaikka ne ovat olleet olemassa jo yli 10 vuotta. WebP-formaatin tulevaisuuteen vaikuttaa siis merkittävästi se, kuinka paljon ja nopeasti sen tukea saadaan laajennettua.

Lähteet

1 Keränen, Vesa; Lamberg, Niko & Penttinen, Jukka. 2003. Julkaisu & kuvankäsittely. Jyväskylä: Docendo Finland.

2 Viljanen, Jarkko; Suvanto, Timo & Karhula, Matti. 2006. Digikuvan peruskirja. Jyväskylä: Docendo Finland.

3 Baxes, Gregory A. 1994. Digital Image Processing. New York: John Wiley & Sons.

4 Grigorik, Ilya. 2017. Image Optimization. Verkkodokumentti. <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization>. 9.2.2017. Luettu 28.2.2017.

5 Patterson, Steve. 2017. The 72 PPI Web Resolution Myth. Verkkodokumentti. <http://www.photoshopessentials.com/essentials/the-72-ppi-web-resolution-myth>. Luettu 28.2.2017.

6 What are JPEG artifacts and what can be done about them. 2012. Verkkodokumentti. StackExchange. <http://photo.stackexchange.com/questions/19270/what-are-jpeg-artifacts-and-what-can-be-done-about-them>. Luettu 1.3.2017.

7 Gonzales, Rafael C & Woods, Richard E. 2008. Digital Image Processing. Upper Saddle River: Pearson Education.

8 Jones, Keith. Pixels, Image Resoution, and Print Sizes. Verkkodokumentti. <http://www.easybasicphotography.com/image-resolution-pixels-print-sizes.html>. Luettu 1.3.2017.

9 Historical trends in the usage of image file formats for websites. 2017. Verkkodokumentti. W3techs Web Technology Surveys. <https://w3techs.com/technologies/history_overview/image_format/all>. 2.3.2017. Luettu 2.3.2017.

10 MIME-types. 2017. Verkkodokumentti. MDN Mozilla Developer Network. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types>. 16.3.2017. Luettu 20.3.2017.

11 Media Types. 2017. Verkkodokumentti. IANA. <http://www.iana.org/assignments/media-types/media-types.xhtml#image>. 7.4.2017. Luettu 20.3.2017.

12 Difference between JPG and JPEG. Verkkodokumentti. Difference between. <http://www.differencebetween.info/difference-between-jpg-and-jpeg>. Luettu 13.3.2017.

13 JPEG2000 vs JPEG (vs TIFF). Verkkodokumentti. Photozone. <http://www.photozone.de/jpeg2000-vs-jpeg-vs-tiff>. Luettu 16.3.2017.

14 Archambault, Michael. 2015. JPEG 2000: The Better Alternative to JPEG That Never Made it Big. Verkkodokumentti. <https://petapixel.com/2015/09/12/jpeg-2000-the-better-alternative-to-jpeg-that-never-made-it-big>. 12.9.2015. Luettu 13.3.2017.

15 Overview of JPEG XR. Verkkodokumentti. JPEG. <https://jpeg.org/jpegxr>. Luettu 13.3.2017.

16 JPEG XR image format. 2017. Verkkodokumentti. Can I use. <https://caniuse.com/#feat=jpegxr>. Luettu 13.3.2017.

17 Niederst Robbins, Jennifer. 2006. Web Design in a Nutshell. Sebastopol: O'Reilly Media.

18 Gelbmann, Metthias. 2013. The PNG image format is now more popular than GIF. Verkkodokumentti. <https://w3techs.com/blog/entry/the_png_image_file_format_is_now_more_popular_than_gif>. 31.1.2013. Luettu 5.3.2017.

19 A basic Introduction to PNG Features. 2009. Verkkodokumentti. libpng.org. <http://www.libpng.org/pub/png/pngintro.html>. 14.3.2009. Luettu 5.3.2017.

20 Portable Network Graphics (PNG) Specification (Second Edition). 2003. Verkkodokumentti. W3C. <https://www.w3.org/TR/PNG>. 10.11.2003. Luettu 5.3.2017.

21 Multiple-image Network Graphics. 2015. Verkkodokumentti. libpng.org. <http://www.libpng.org/pub/mng>. 11.1.2015. Luettu 6.3.2017.

22 Mobile Optimization. 2016. Verkkodokumentti. Moz. <https://moz.com/learn/seo/mobile-optimization>. Luettu 6.3.2017.

23 Grigorik, Ilya. 2017. Optimizing Encoding and Transfer Size of Text-Based Assets. Verkkodokumentti. <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/optimize-encoding-and-transfer>. 9.2.2017. Luettu 8.3.2017.

24 Grigorik, Ilya. 2017. HTTP Caching. Verkkodokumentti. <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching>. 9.2.2017. Luettu 8.3.2017.

25 Interesting stats. Verkkodokumentti. HTTP Archive. <http://httparchive.org/interesting.php>. Luettu 17.3.2017.

26 HTTP Overview. 2017. Verkkodokumentti. Tutorialspoint. <https://www.tutorialspoint.com/http/http_overview.htm>. Luettu 15.3.2017.

27 How Mobile is Changing Business. 2013. Verkkodokumentti. Kissmetrics Blog. <https://blog.kissmetrics.com/mobile-is-changing-business>. Luettu 12.3.2017.

28 Cutts, Matt & Singhal, Amit. 2010. Verkkodokumentti. Using site speed in web search ranking. 9.4.2010. Luettu 13.3.2017.

29 A new image format for the Web. 2016. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp>. 4.3.2016. Luettu 5.2.2017.

30 Frequently Asked Questions. 2016. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/faq>. 21.12.2016. Luettu 5.2.2017.

31 Compression Techniques. 2016. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/docs/compression>. 21.12.2016. Luettu 5.2.2017.

32 WebP, a new image format for the Web. 2010. Verkkodokumentti. Chromium Blog. < https://blog.chromium.org/2010/09/webp-new-image-format-for-web.html>. 30.9.2010. Luettu 5.2.2017.

33 Inside WebM Technology: VP8 Intra and Inter Prediction. 2010. Verkkodokumentti. The WebM Project. <http://blog.webmproject.org/2010/07/inside-webm-technology-vp8-intra-and.html>. 20.7.2010. Luettu 14.2.2017.

34 Mynttinen, Timo. 2009. Intra-frame DCT -koodaus. Verkkodokumentti. <cna.mikkeliamk.fi/Public/MynttinenTimo/Digitaalinen%20informaatio/MPEG%20videopakkaus/Intra-frame%20DCT%20-koodaus.ppt>. 24.8.2009. Luettu 26.2.2017.

35 Honkanen, H. Kuvantallennusformaatit. Verkkodokumentti. <http://gallia.kajak.fi/opmateriaalit/yleinen/honHar/ma/TVMonitor_MPEG-standardit.pdf>. Luettu 26.2.2017.

36 WebP Container Specification. 2016. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/docs/riff_container>. 21.12.2016. Luettu 1.3.2017.

37 Resource Interchange File Format (RIFF). 2017. Verkkodokumentti. Microsoft. <https://msdn.microsoft.com/en-us/library/windows/desktop/ee415713(v=vs.85).aspx>. Luettu 1.3.2017.

38 Precompiled Utilities. 2017. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/docs/precompiled>. 31.1.2017. Luettu 2.3.2017.

39 Getting started. 2016. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/docs/using>. 21.12.2016. Luettu 6.3.2017.

40 WebP image format. 2017. Verkkodokumentti. Can I use. <https://caniuse.com/#feat=webp>. Luettu 5.3.2017.

41 Desktop vs Mobile vs Tablet Market Share Worldwide. 2017. Verkkodokumentti. Statcounter. <http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet>. Luettu 13.3.2017.

42 Johansson, Johan. 2013. How To Make Your Websites Faster On Mobile Devices. Verkkodokumentti. 3.4.2013. Luettu 13.3.2017.

43 Mobile Browser Market Share Worldwide. 2017. Verkkodokumentti. StatCounter. <http://gs.statcounter.com/browser-market-share/mobile/worldwide/#monthly-201703-201703-bar>. Luettu 13.3.2017.

44 What is the WebP Codec for Windows? 2015. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/docs/webp_codec>. 19.12.2015. Luettu 14.3.2017.

45 Webmproject/libwebp releases. 2017. Verkkodokumentti. GitHub. <https://github.com/webmproject/libwebp/releases>. 31.1.2017. Luettu 14.3.2017.

46 Shankland, Stephen. 2013. Mozilla takes a fresh look at Google's WebP image format. Verkkodokumentti. <https://www.cnet.com/news/mozilla-takes-a-fresh-look-at-googles-webp-image-format>. 8.4.2013. Luettu 18.3.2017.

47 Brinkmann, Martin. 2016. Mozilla plans to add Webp support to Firefox. Verkkodokumentti. <http://www.ghacks.net/2016/08/24/mozilla-webp-support-firefox>. 24.8.2016. Luettu 18.3.2017.

48 Shankland, Stephen. 2016. Apple tests Google graphics format to speed up websites. Verkkodokumentti. <https://www.cnet.com/news/apple-ios-macos-tests-googles-webp-graphics-to-speed-up-web>. 19.7.2016. Luettu 18.3.2017.

49 Paul, Ryan. 2011. Mozilla rejects WebP image format, Google adds it to Picasa. Verkkodokumentti. <https://arstechnica.com/information-technology/2011/05/mozilla-rejects-webp-image-format-google-adds-it-to-picasa>. 24.5.2011. Luettu 18.3.2017.

50 Browser Market Share Worldwide. 2017. Verkkodokumentti. StatCounter. <http://gs.statcounter.com/browser-market-share>. Luettu 18.3.2017.

51 Usage of WebP for websites. 2017. Verkkodokumentti. W3techs Web Technology Surveys. <https://w3techs.com/technologies/details/im-webp/all/all>. 19.3.2017. Luettu 19.3.2017.

52 Structural Similarity Index (SSIM) for measuring image quality. 2017. Verkkodokumentti. MathWorks. <https://se.mathworks.com/help/images/ref/ssim.html>. Luettu 20.3.2017.

53 Bovik, Alan Conrad; Sheikh, Hamid Rahim; Simoncelli, Eero P & Wang, Zhou. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. Verkkodokumentti. <http://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf>. 4.4.2004. Luettu 20.3.2017.

54 Bovik, Alan Conrad & Li, Chaofeng. 2010. Content-weighted video quality assessment using a three-component image model. Verkkodokumentti. <http://live.ece.utexas.edu/publications/2010/li_jei_jan10.pdf>. 7.1.2010. Luettu 20.3.2017.

55 WebP Compression Study. 2016. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/docs/webp_study>. 21.12.2016. Luettu 27.2.2017.

56 Rowse, Darren. 2017. RAW vs JPEG. Verkkodokumentti. <https://digital-photography-school.com/raw-vs-jpeg>. Luettu 28.2.2017.

57 Nikon Electronic Format (NEF). 2017. Verkkodokumentti. Nikon. <http://www.nikonusa.com/en/learn-and-explore/a/products-and-innovation/nikon-electronic-format-nef.html>. Luettu 28.2.2017.

58 Resize images. 2017. Verkkodokumentti. Adobe Support. <https://helpx.adobe.com/photoshop/using/resizing-image.html>. 15.2.2017. Luettu 1.3.2017.

59 MATLAB. 2017. Verkkodokumentti. MathWorks. <https://se.mathworks.com/products/matlab.html>. Luettu 30.3.2017.

60 Save files in graphics formats: Save in JPEG format. 2017. Verkkodokumentti. Adobe Support. <https://helpx.adobe.com/photoshop/using/saving-files-graphics-formats.html>. 15.2.2017. Luettu 17.2.2017.

61 Sutton, Zach. 2013. AdobeRGB vs. sRGB. Verkkodokumentti. <https://fstoppers.com/pictures/adobergb-vs-srgb-3167>. 17.2.2013. Luettu 19.2.2017.

62 Cwebp. 2017. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/docs/cwebp>. 31.1.2017. Luettu 16.3.2017.

63 Gif2webp. 2017. Verkkodokumentti. WebP. <https://developers.google.com/speed/webp/docs/gif2webp>. 31.1.2017. Luettu 10.3.2017.

64 About PageSpeed Insights. 2015. Verkkodokumentti. Google PageSpeed Tools. <https://developers.google.com/speed/docs/insights/about>. 27.5.2015. Luettu 29.3.2017.

65 YSlow. Verkkodokumentti. YSlow. <http://yslow.org/>. Luettu 29.3.2017.